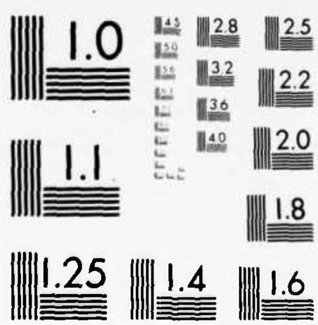
AD-A039 819 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA PROGRAM --ETC F/G 15/4
DOCUMENTATION OF CURRENT IDA COMPUTER MATERIAL DEVELOPED FOR DC--ETC(U)
JAN 77 L A SCHMIDT DCPA01-76-C-0213
UNCLASSIFIED IDA/HQ-77-19226 NL

1 OF 3
ADA039 819

039



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Copy 3 of 14 copies

4

DOCUMENTATION OF CURRENT IDA COMPUTER MATERIAL DEVELOPED FOR DCPA

Volume II

Leo A. Schmidt

January 1977

INSTITUTE FOR DEFENSE ANALYSES
Program Analysis Division

Contract No. DCPA01-76-C-0213
Work Unit 4126G

IDA Log No. HQ 77-19226

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

The work reported in this document was conducted under Contract No. DCPA01-76-C-0213 for the Defense Civil Preparedness Agency. The publication of this document does not indicate endorsement by the Defense Civil Preparedness Agency, nor should the contents be construed as reflecting the official position of that agency.

DOCUMENTATION OF CURRENT IDA COMPUTER MATERIAL
DEVELOPED FOR DCPA

Volume II

by

Leo A. Schmidt

for

Defense Civil Preparedness Agency
Washington, D.C. 20301

DCPA REVIEW NOTICE

This report has been reviewed in the Defense Civil Preparedness Agency and approved for publication. Approval does not signify that the contents necessarily reflect the views and policies of the Defense Civil Preparedness Agency.

INSTITUTE FOR DEFENSE ANALYSES
Program Analysis Division
400 Army-Navy Drive, Arlington, Virginia 22202

Contract No. DCPA01-76-C-0213
Work Unit 4126G

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NA	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Documentation of Current IDA Computer Material Developed for DCPA. Volume II.		5. TYPE OF REPORT & PERIOD COVERED NA
		6. PERFORMING ORG. REPORT NUMBER NA
7. AUTHOR(s) (10) Leo A. Schmidt		8. CONTRACT OR GRANT NUMBER(s) (15) DCPA01-76-C-0213 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS INSTITUTE FOR DEFENSE ANALYSES Program Analysis Division 400 Army-Navy Drive, Arlington, VA 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit 4126G
11. CONTROLLING OFFICE NAME AND ADDRESS DEFENSE CIVIL PREPAREDNESS AGENCY Washington, D.C. 20301		12. REPORT DATE (11) January 1977
		13. NUMBER OF PAGES Vol. II--276 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 239p.		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(18) IDA/HQ

18. SUPPLEMENTARY NOTES

(19) 77-19226

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Damage Assessment

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper is a documentation of computer materials developed by the Institute for Defense Analyses (IDA) for use by the Defense Civil Preparedness Agency (DCPA). All IDA physical data processing materials (IBM cards, magnetic tape, computer printouts) have been surveyed and catalogued. All computer programs are written in FORTRAN (a general knowledge of this language is assumed in the detailed descriptions contained herein). Computer programs considered useful by IDA have

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403219

1/B

20. continued

been included and documented. A group of general purpose subprograms are described, along with their interfaces with the using programs. Data file formats also have been developed, along with programs for managing these files. Such programs and resulting files are described in detail.

CONTENTS

Volume II

SUBPROGRAMS

This volume contains documentation of standardized FORTRAN subprograms. The documentation for these subprograms was initiated in 1972, when many of the basic subprograms from a number of computer programs were put into a single system. The acronym NEVUNS is used to identify this set of subprograms.

Two basic types of subprograms are in this system, NEVUNS Standard and NEVUNS Compatible. The former have the standard documentation and are, hopefully, carefully tested. The latter have the same external communication (through block common) but may have less full documentation. The standard subprograms are intended to accomplish a specific purpose which could be used in a variety of main programs, whereas the compatible subprograms represent subprograms taken from some existing program because they might have use elsewhere.

The documentation for a NEVUNS Standard subprogram consists of:

A. General--this section is a general description of the subprogram including intended use and general method of approach.

B. Requirements on Calling Program.

The parameters which must be set by the calling program are defined and the external communication described. The external communication is usually through block common. An attempt has been made to group in a common block similar types of variables, thus the same common blocks may be used in several subprograms but some variables may not be used in a particular subprogram.

Only those in use in a particular subprogram are defined in its documentation. If a subprogram requires special features from a using program, such as initializing calls, these must be specified here.

C. Algorithm Implemented

This section describes the type of calculation made in the subprogram. It describes the calculation in mathematical language and defines the internal subprogram variables when necessary to readily interpret the subprogram. If helpful to understand, this section may contain flow charts or excerpts from other documents.

D. Program Listing

A listing of the subprogram forms this section. A subprogram source deck should be filed in the IDA Civil Defense Card Deck Library which is identical to this listing.

SUBROUTINE CTYDAM (DECK #1)

A. GENERAL

This subroutine does calculations for individual cities as part of a blast optimization package.[1] It uses the square root damage law to determine the payoff for individual weapons as a function of the number of weapons attacking ~~the~~ a city. As the weapons are added the weapon payoff is placed in a storage array, with the city ID placed in another storage array. The process is continued until the weapon payoff drops below some input value of marginal return when the process is terminated. After all cities have been treated, all the weapons ~~are~~ CAN BE ordered in payoff and those with highest payoff are selected for ~~the~~ a final ~~use~~ OPTIMIZED ALLOCATION.

The subroutine includes a terminal ABM price option. When this option is exercised all weapons up to a specified number N are given the same value of pay. The number is that number which maximizes average pay, where the pay is computed by the square root damage law minus the price. The pay assigned is this maximum average payoff. Of course, no weapons are assigned if the maximum average payoff is less than the minimum marginal return. After these initial weapons are assigned the optimization proceeds as usual.

A special procedure is used for cities described by only a few tracts where the square root damage law would not be expected

---

[1] The methodology is described in some detail in IDA Study S-394, "Methodologies for Evaluating the Vulnerability of National Systems," Volume I, by J. McGill, et al, June 1972.

to apply. For the larger cities an option allows varying the efficiency of the weapon allocation, through the parameter ALPHR, as a function of city size, weapon CEP, lethal radius, and CEP.[2] Another special procedure is used if a dispersion option is exercised where the population is considered uniform over a ring.

B. REQUIREMENTS ON THE CALLING PROGRAM

The communication with the calling program is through block common. It is assumed that data for all the cities within a county are available simultaneously in the common block/CITYAR/. The parameters which must be defined by the calling program are those in the following common blocks:

/COUNPR/	JCOVER	-	an indicator which is one if the county is covered by Area ABM, and zero otherwise. It is used as a ploy to force bypassing a county if JAAVD is 1. Otherwise it is ignored.
/CITYAR/	IDCTY(K)	-	a city identifier - equal to 100000000 x Area ABM city No + 10000 x County ID No + City ID No
	CPRICA(K)	-	the terminal ABM price
	SIGXYC(K)	-	product of NS and EW standard deviation of the city population
	VALC(K)	-	the total city value - used as a multiplie of the % payoff per weapon
	NTRCTS(K)	-	the number of data tracts in the city
	NMBCTY	-	the number of cities in the county

---

2. The justification for this is contained in L. A. Schmidt, "A Sensitivity Analysis of Blast Fatality Calculations, IDA Paper P-762, ~~(date)~~ JANUARY 1971.

/WPNPR/	DEL	- the weapon reliability
	CEP	- the weapon aiming error
	CRTYLW	- the cube root of the weapon yield
/VULPR/	RLONE	- lethal radius for a 1 MT weapon
	RLSONE	- the square of RLONE
/TMPALG/	JTABM	- if one includes terminal ABM in optimization, otherwise not
	JAAUD	- if one, do not attack if covered by area ABM, i.e., JCOVERY
	KRTAD	- if 1, vary ALPHR with city and weapon characteristics; 2, set ALPHR = 2.
	SMLVAL	- minimum value of marginal return with a city not treated by square root law.
		- ~~minimum value of marginal return with a city treated by square root law~~
	SMLRVL	- minimum value of marginal return for cities treated by square root law
	KDISPN	- if one, use dispersion option
	DISMIN	- minimum value (population) of city to use dispersion option
	KDTGT	- if one, retarget against dispersed population
	~~D~~	- if zero, target against undispersed population and evaluate against dispersed population
	DISRAD	- the radius of the dispersed population
	KDSML	- if one, use dispersion option on small cities
	KRTDAP	- formula to select α in dispersion calculations = 0 use α = 0 = 2 use input value
	ALPHDR	- input value

The output of the subroutines are placed in the ~~array~~ COMMON BLOCK /ORDOPT/. As each weapon is added the counter LWPCT is incremented, the weapon pay is stored in PAYW and city ID is IDCIT under the index.

C. ALGORITHMS IMPLEMENTED

For simplicity, the changes with the dispersion option will be summarized after the non dispersion calculations are described.

If the area defense avoidance switch, JAAVD,is on and the site coverage switch, JCOVER, is on, the subroutine is exited.

The weapon lethal radius is converted to the actual yield by multiplying by the square root of the yield. If the terminal defense calculation switch is off, this is signaled by setting the city defense price to zero. If the terminal defense avoidance switch is on and the price is not zero, this city is bypassed.

If this is only a single tract the following special calculation is performed:

An expected return T1 is calculated by

$$T1 = \frac{DEL \cdot VALC(R)}{1 + \frac{CPRICE}{DEL}} .$$

If this return is less than the small marginal return SMLVAL, the city is bypassed. If not, a number of weapons equal to 1 + CPRICE(K)/DEL is added. (In the absence of ABM, this is one weapon.) To add a weapon ~~this~~ THE counter LWPCT is incremented, the pay is set equal to T1 IN THE ARRAY PAYW, and the city ID is given IN THE ARRAY IDCIT. If enough city value remains so the expected return is greater than SMLVAL this process is repeated until this condition does not apply.

If there is more than one tract an expected return from a single weapon aimed at the center of the city is calculated by

$$RET = \frac{DEL \cdot VALC(K)}{1 + \frac{1.386\ SIGXYC(K)}{RLSQ}} .$$

This assumes both the weapon kill probability distance and city population are circular gaussian.

If the return is greater than 2 x the minimum return, SMLVAR, the square root law is used. Otherwise, the previously described procedure is used to add weapons.

For square root law calculations if KRSTAD is not 1, ALPDR is set to 2. If it is one the ALPDR is calculated by

$$\text{ALPDR} = 2.89/\{2.906-0.66\cdot\text{DEL}+0.82(.2+0.29(\frac{\text{CEP}}{\text{RL}})^2)$$

$$+\ 0.81\ \log_{10}(\frac{\text{RL}}{\sqrt{\text{SIGXYC(K)}}})\}\ .$$

A value of K is computed by

$$K = \frac{\text{RLSQ}\cdot\text{DEL}\cdot\text{ALPDR}}{\text{SIGXYC(K)}}$$

As a function of number of weapons added, NW the value surviving S, is computed by

$$S = \text{VALC(K)}\ \exp(-\sqrt{x})(1+\sqrt{x})$$

where

$$x = K\cdot\text{NW}.$$

If there is no ABM, i.e., CPRICE(K) is zero, weapons are added until the weapon pay drops below SMLRVL.

If ABM is present the average return from NW weapons is computed by the total pay (VALC(K)•S) from the square root law with N-CPRICE(K) weapons. Weapons are added until the average return reaches a maximum. At this point, if the average return is greater than SMLRVL, all N weapons are added at a value equal to this average return.

In ~~this~~ THE later ordering by weapon value this group of weapons will preserve its identity as a group since they all have the same pay. After this group of weapons are added, succeeding weapons are added AS if no ABM were present until the payoff falls below the minimum return SMLRVL.

The dispersion option is exercised if the switch KDISPN is on. for cities above DISMIN in size. If switch KDSML is off the option is only used for those cities treated by the square root law.

For the small cities with all weapons aimed at the center

let

$$R_S = \frac{{D_R}^2 \cdot {R_L}^2}{\mathrm{SIGXYC(K)}}$$

where

R_L is the weapon lethal radius

D_R is the dispersion radius.

Then the fraction destroyed by a single weapon is given by

$$f = \frac{1}{R_S}(1-\exp(-R_S))$$

The rest of the small city calculation is the same.

For the square root law calculation the method of deriving the square root law obtains a weapon density as a function of position. This may be evaluated against population in a uniform disk. One obtains for fraction of fatalities

$$f = \begin{cases} \frac{1}{\tau}[\sqrt{x} + e^{-\sqrt{x}} - 1] & x \leq \tau^2 \\ \frac{1}{\tau}[\tau + e^{-\sqrt{x}}(1-e^{\tau})] & x > \tau^2 \end{cases}$$

where

$$\tau = \frac{R_D^{\,2}}{2 \cdot SIGXYC(K)}$$

AND

x is as calculated before.

In calculation x the following values of α may be used

KRTDAD = 0 $\alpha = 2$

= 1 $\alpha = 1+DEL$

= 2 α = input value ALP4DR

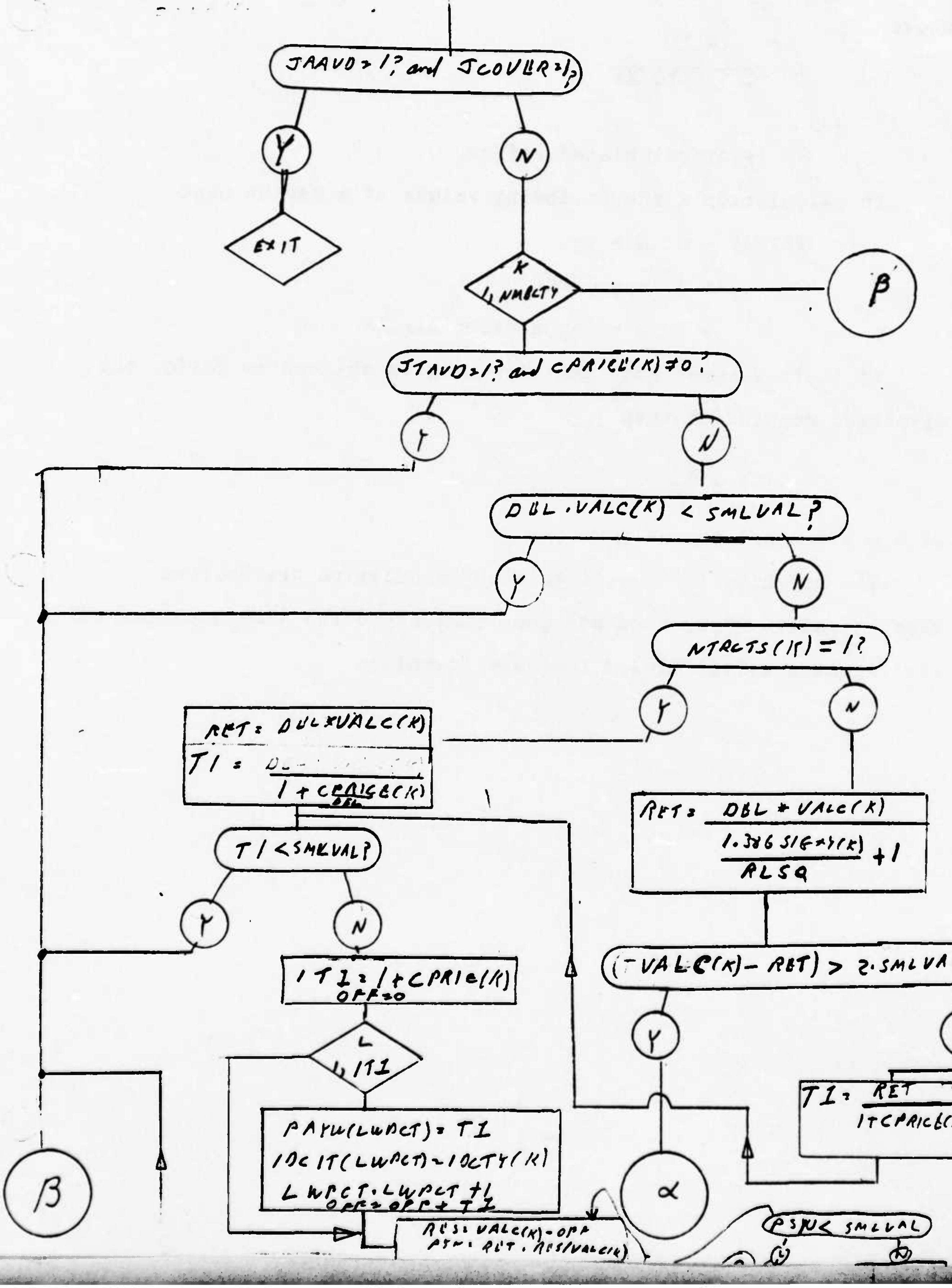
If it is assumed that the targeting is shifted to follow the dispersed population then

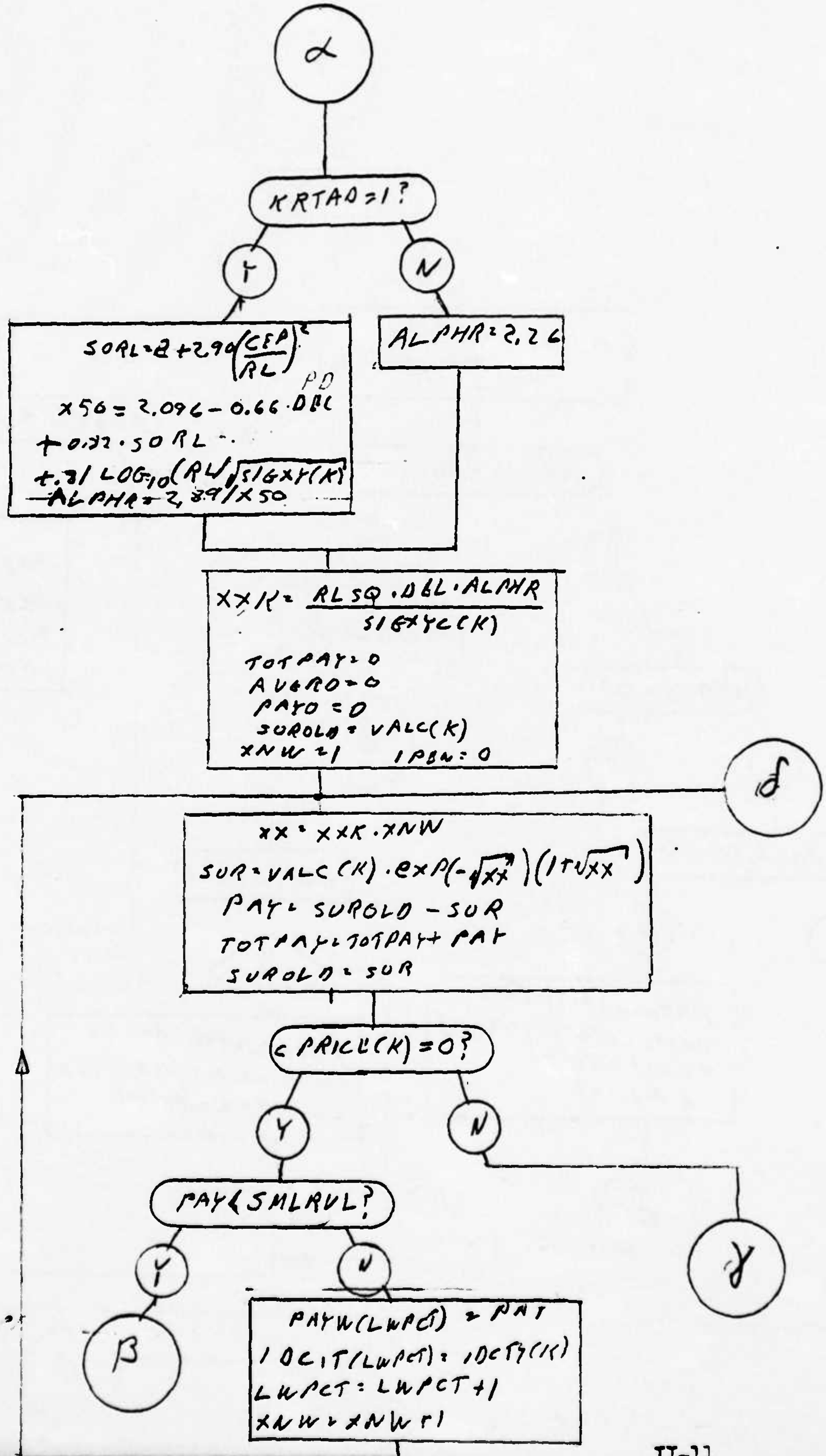
$$f = \frac{x}{2\tau}$$

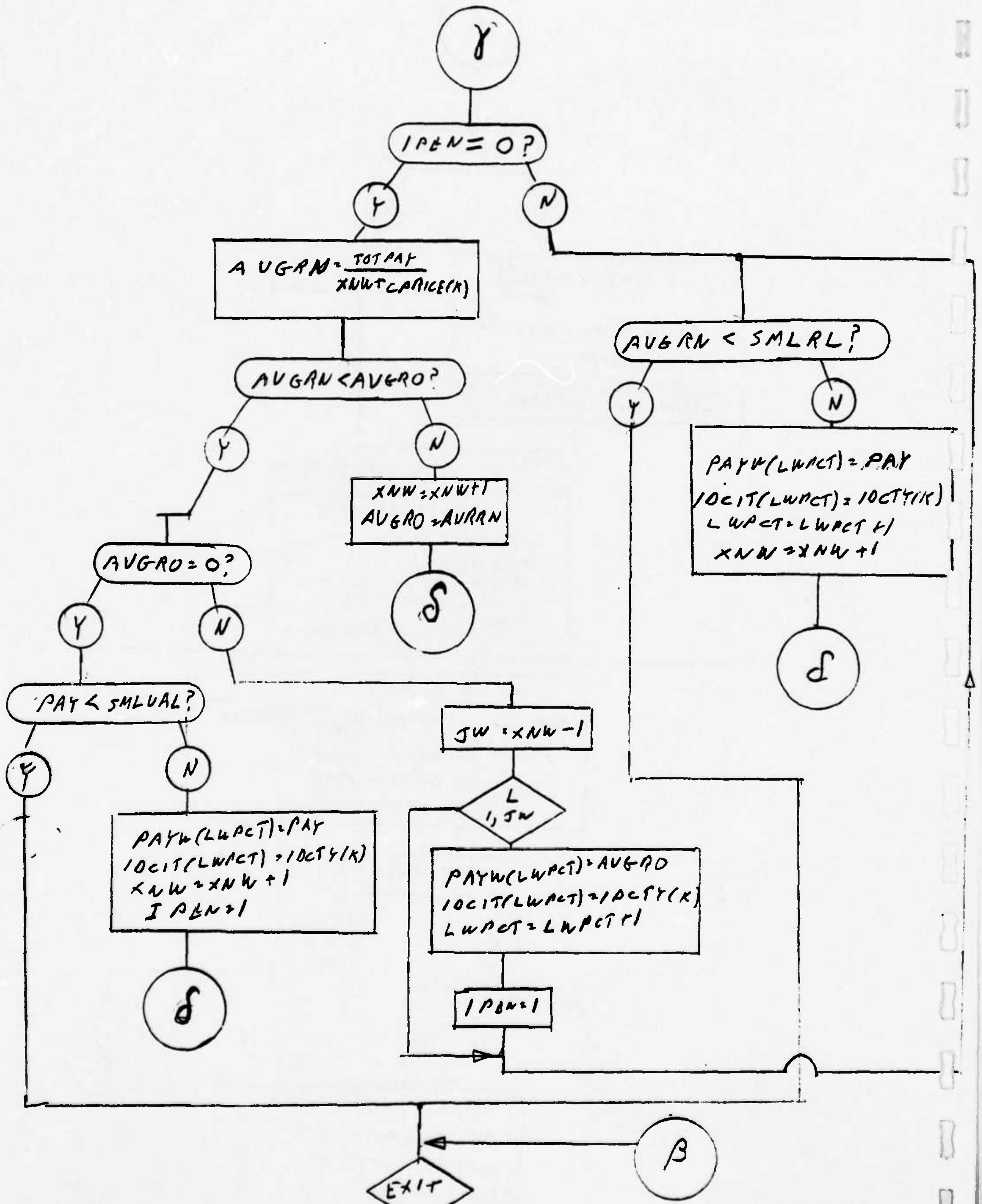
with α set equal to one.

The formulas for the dispersion calculation are derived theoretically. They have not been subject to the same experimental verification as the square root law formulas.

SUBROUTINE CTYDAM




α
KRTAD=1?
Y
N
SORL=2+2.90(CEP/RL)²
X50=2.096-0.66·DBL
+0.32·SORL
+.31 LOG10(RL/√SIGXY(K))
ALPHR=2.39/X50
ALPHR=2.26
XXK=RLSQ·DBL·ALPHR/SIGXYC(K)
TOTPAY=0
AVGRO=0
PAYO=0
SUROLD=VALC(K)
XNW=1
IPBN=0
δ
XX=XXK·XNW
SUR=VALC(K)·EXP(-√XX)(1+√XX)
PAY=SUROLD-SUR
TOTPAY=TOTPAY+PAY
SUROLD=SUR
CPRICE(K)=0?
Y
N
PAY<SMLRVL?
Y
N
γ
β
PAYW(LWPCT)=PAY
IDCIT(LWPCT)=IDCTY(K)
LWPCT=LWPCT+1
XNW=XNW+1


γ
IPEN = 0?
Y
N
AUGRN = TOTPAY / XNWT CAPRICE(K)
AUGRN < AUGRO?
Y
N
XNW = XNW + 1
AUGRO = AUGRN
δ
AUGRO = 0?
Y
N
PAY < SMLVAL?
Y
N
PAYW(LWPCT) = PAY
IOCIT(LWPCT) = IOCTY(K)
XNW = XNW + 1
IPEN = 1
δ
JW = XNW - 1
L
1, JW
PAYW(LWPCT) = AUGRO
IOCIT(LWPCT) = IOCTY(K)
LWPCT = LWPCT + 1
IPEN = 1
AUGRN < SMLRL?
Y
N
PAYW(LWPCT) = PAY
IOCIT(LWPCT) = IOCTY(K)
LWPCT = LWPCT + 1
XNW = XNW + 1
δ
β
EXIT

```
                SUBROUTINE CTYDAM
          C     NEVUNS STANDARD
          C     LAST REVISED NOV. 23, 1972

          C     ALLOCATES WEAPONS TO EACH CITY USING SQUARE ROOT LAW UNTIL THE
          C     WEAPON RETURN FALLS BELOW SOME SPECIFIED VALUE. INCLUDES
          C     TERMINAL ABM CAPABILITY AND SPECIAL ROUTINES FOR SMALL CITIES.
          C     SMALL CITY ROUTINES DIFFERENT FROM ALLEGRO BECAUSE MORE THAN
          C     ONE WEAPON IS ALLOWED ON SMALL CITIES.

                DIMENSION CPRICE(20)
                COMMON/TMPALG/ BLK,JTABM,BLY,JAAVD,JTAVD,BLZ,KRTAD,SMLVAL,SMLRVL,
               1 BLN(18),KDISPN,DISMIN,KDTGT,DISRAD,KDSML,KRTDAD,ALPHOR,BLX(8)
                COMMON/DROOPT/LWPCT, PAYW(6000),IOCIT(6000),BLG(6000)
                COMMON/CITYAR/BLO(300),IOCTY(20),CPRICA(20),BLE(20),SIGXYC(20),
               1 VALC(20), BLF(20),NTRCTS(20),NMBCTY
                COMMON/COUNPR/ BLH(30),JCOVER,BLI(4)
                COMMON/WPNPR/BLA(2),DEL,CEP,BLB(20),CRTYLW,BLJ
                COMMON/VULPR/BLC(107), RLONE,RLSONE

                IF(JAAVD.EQ.1.AND.JCOVER.EQ.1)GO TO 8
                GO TO 9

          C....BYPASS THIS ON AREA DEFENSE AVOIDANCE OPTION

              8 RETURN
              9 CONTINUE

                RL   = RLONE*CRTYLW
                RLSQ = RLSONE*CRTYLW
                DO 10 K = 1,NMBCTY

          C....BYPASS CITY ON TERMINAL   DEFENSE AVOIDANCE OPTION

                CPRICE(K) = CPRICA(K)
                IF ( JTABM .NE. 1) CPRICE(K) = 0.
                IF(JTAVD.EQ.1.AND.CPRICE(K).NE.0.) GO TO 10

          C....CITY VALUE TOO  SMALL TO BE ATTACKED

                IF((DEL* VALC(K)).LT.SMLVAL) GO TO 10
                IF(NTRCTS(K).NE.1) GO TO 11

          C....SINGLE TRACT CITY ASSUM ONE EXPLODING BOMB GETS ALL THE VALUE

                RET = DEL*VALC(K)
                T1 = RET/(1. + CPRICE(K)/DEL)
             14 CONTINUE
                IF(T1.LT.SMLVAL) GO TO 10
```

```
          ITI=1.+CPRICE(K)/DEL
          OFF = 0.
          DO 12 L = 1,IT1
          LWPCT=LWPCT+1
          PAYW(LWPCT)=T1
          IDCIT(LWPCT)=IDCTY(K)
          OFF =  OFF + TI
       12 CONTINUE
   13     CONTINUE
          RES = VALC(K) - OFF
          PYN = RET*RES/VALC(K)
          IF(PYN .LT. SMLVAL) GO TO 10
          LWPCT = LWPCT + I
          PAYW(LWPCT) = PYN
          IDCIT(LWPCT) = IDCTY(K)
          OFF = OFF + PYN
          GO TO 13

       11 CONTINUE
   C...   ASSUME CITY SO SMALL ONLY ONE BOMB IN THE CENTER IS USED
          IF( KDISPN .EQ. 1) GO TO 60
   6I     CONTINUE
          RET=DEL*VALC(K)/(1.386*SIGXYC(K)/RLSQ+1.)
   62     CONTINUE

   C....IF LESS THAN 1/2 VALUE FROM ONE BOMB IS SQ. ROOT LAW.

          IF((VALC(K)-RET).GT.2.*SMLVAL) GO TO 20
          TI=RET/(1.+CPRICE(K))
          GO TO 14
   60     CONTINUE
   C... ONE BOMB AGAINST DISPERSED POPULATION
          IF( KDSML .NE. 1) GO TO 61
          IF( VALC(K) .LT. DISMIN) GO TO 61
          RSRLS = DISRAD*DISRAD*RLSQ/SIGXYC(K)
          FT = (1. - EXP(-RSRLS))/RSRLS
          RET = DEL*VALC(K)*FT
          GO TO 62


       20 CONTINUE
          IF( KDISPN .EQ. I) GO TO 70
   71     CONTINUE
   C...., NOW USE SQRT LAW
          IF(KRTAD.EQ.1) GO TO 21
   C...*******....**********
          ALPHR = 2.0
   C...*******....**********
          GO TO 22
       21 CONTINUE
   C....INSERT ADJUSTMENTS HERE TO ALPHR
          CEORL=CEP/RL
          SORL=.2+.290*CEORL*CEORL
          SXYRT=SQRT(SIGXYC(K))
          X50=2.906-0.66*DEL+0.82*SORL+0.81*ALOG10(RL/SXYR!)
```

```
              ALPHR=2.89/X50
              GO TO 22
        70    CONTINUE
        C... OISPERSION CALCULATION WITH ORIGINAL TARGETING ON GAUSSIAN POPN.
              IF( VALC(K) .LT. DISMIN) GO TO 71
              TAU = OISRAO*DISRAD/2.
              IF( KDTGT .EQ. 1) GO TO 72
              TAUS = TAU*TAU
              EXPT = EXP(TAU)
              IF( KRTOAD .NE. 0) GO TO 73
              ALPHR = 2.0
              GO TO 22
        73    CONTINUE
              IF( KRTDAO .NE. 1) GO TO 74
        C...., BETTER VALUES OF ALPHR MIGHT  BE USEO
              ALPHR = 1. + OEL
              GO TO 22
        74    CONTINUE
              ALPHR = ALPHOR
              GO TO 22
        72    CONTINUE
        C... SHIFT TARGETING
              ALPHR = 1.
              GO TO 22

           22  CONTINUE
              XXK=RLSQ*DEL*ALPHR/SIGXYC(K)
              TOTPAY=0.
              AVGRO=0.
              XNW=1.
              SUROLD=VALC(K)
              IPEN=0

           25 CONTINUE

        C.... NOW NEW VALUE OF NO. OF WEAPONS

              XX=XXK*XNW
              SXX=SQRT(XX)
              IF ( KDISPN .EQ. 1) GO TO 80
        81    CONTINUE
              SUR=VALC(K)*EXP(-SXX)*(1.+SXX)
        90    CONTINUE
              PAY=SUROLD-SUR
              TOTPAY=TOTPAY+PAY
              SUROLO=SUR
              IF(CPRICE(K).NE.0.) GO TO 30
              IF(PAY.LT.SMLRVL) GO TO 10

        C.... ADD WEAPON TO LIST

              LWPCT=LWPCT+1
              PAYW(LWPCT)=PAY
              IDCIT(LWPCT)=IDCTY(K)
              XNW=XNW+1.
```

```
      GO TO 25
C     SQUARE ROOT LAW DISPERSION CALCULATION
80    CONTINUE
      IF ( VALC(K) .LT. DISMIN) GO TO 81
      IF( KDTGT .EQ. 1) GO TO 82
      IF( XX .GT. TAUS) GO TO 83
      FT = (SXX + EXP(-SXX)*( 1.- EXPT))/TAU
      SUR =  VALC(K) *(1. - FT)
      GO TO 90
83    CONTINUE
      FT = (TAU + EXP(-SXX)*(1. - EXPT))/TAU
      SUR = VALC(K) *(1. - FT)
      GO TO 90
82    CONTINUE
      SUR = VALC(K)* (1. - XX*ALPHR)/(2.*TAU)
      GO TO 90

   30 CONTINUE

C....TERMINAL  ABM FROM HERE ON.  ASSUMES A STRICT PRICE MODEL

      IF(IPEN.NE.0) GO TO 50

C.... CONTINUE TILL AVGRN STARTS DECREASING

      AVGRN=TOTPAY/(XNW+CPRICE(K)/DEL)
      IF(AVGRN.LT.AVGRO)GO TO 35
      XNW=XNW+1.
      AVGRO=AVGRN
      GO TO 25

   35 CONTINUE
      IF(AVGRO.NE.0.) GO TO 40

C.... ADD ONE WEAPON

      IF(PAY.LT.SMLRVL)GO TO 10
      LWPCT=LWPCT+1
      PAYW(LWPCT)=PAY
      IDCIT(LWPCT)=IDCTY(K)
      XNW=XNW+1.
      IPEN=1
      GO TO 25

   40 CONTINUE

C....ADD WEAPONS ALL AT AVERAGE RETURN

      JW = XNW - 1. + CPRICE(K)/DEL
      DO 41 IL =1,JW
      LWPCT=LWPCT+1
      PAYW(LWPCT)=AVGRO
      IDCIT(LWPCT)=IDCTY(K)
```

```
      41 CONTINUE
         IPEN=1
         GO TO 50
      50 CONTINUE
C....KEEP ADDING WEAPONS UNTIL MARGIN RETURNS TO SMALL
         IF( PAY .LT.SMLRVL) GO TO 10
         LWPCT=LWPCT+1
         PAYW(LWPCT)=PAY
         IDCIT(LWPCT)=IDCTY(K)
         XNW=XNW+1.
         GO TO 25
      10 CONTINUE
         RETURN
         END
```

SUBROUTINE NEWCTY (DECK #2)

A. GENERAL

This subroutine reads values of latitude, longitude, and population for individual tracts of a city and places them in the storage array ST44TA. The center of gravity of the population is computed and distances are converted into miles NS and EW from the center of gravity. The standard deviation of the population are computed.

B. REQUIREMENTS ON CALLING PROGRAM

The subroutine assumes data in standard tract form in an input medium MA. The file format is CTRTA.

The calling program must supply the variable NNEND in common block /TMPAND/. If the city name is this value the subroutine exits without further action.

The subroutine supplies statistical values to the common block /CITYPR/ and /ST44TA/.

The following variables are supplied in CITYPR by the subroutin

NMCIT	city name
CTLAT	latitude of CG of population
CTLON	longitude of CG of population
CTCLAT	cos of latitude of CG of population
CITSX	standard deviation of EW component of population
CITSY	standard deviation of NS component of population
NTRCTS	number of tracts in city

In the array ST~~HH~~44TA the subroutine places

X(I)	distance East of CG of tract (nmi)
Y(I)	distance NORTH ~~West~~ of CG of tract (nmi)
POP(I)	population of tract
V(I)	population of tract.

C. ALGORITHMS IMPLEMENTED

A first record is read in 2DA4 ~~OAH~~ formal if the first four SYMBOLS ~~values~~ equal the VARIABLE NNEND the subroutine sets the variable IPASS in the calling sequence to one and exits. Otherwise the variable IPASS is set equal to zero and the following actions are carried out. The latitude, longitude, and population of successive tracts are read and placed in storage arrays Y, X, and POP. This is continued until a negative latitude is encountered which terminates the reading. The number of tracts, NTRCTS, is counted.

The total population is obtained from

$$\mathrm{TOTPOP} = \sum_{I=1}^{\mathrm{NTRCTS}} \mathrm{POP(I)}$$

The center of gravity of population is computed by

$$\mathrm{FLA\cancel{N}TC} = \frac{\sum_{i=1}^{\mathrm{NTRCTS}} \mathrm{Y(I)POP(I)}}{\mathrm{TOTPOP}}$$

$$\mathrm{FLONC} = \frac{\sum_{i=1}^{\mathrm{NTRCTS}} \mathrm{X(I)\cdot POP(I)}}{\mathrm{TOTPOP}} .$$

The array V(I) is set equal to POP(I).

The units are converted to nautical miles on a square grid by

$$Y(I) = (Y(I) - FLATC) \cdot 60$$

$$X(I) = (FLONGC - X(I)) \cdot 60 \cdot FLFCT$$

where

$$FLFCTC \cos (FLATC) \ .$$

Finally, the standard deviations in the East-West direction are computed by

$$\sigma_x^2 = \frac{\sum_{I=1}^{NTRCTS} X(I)^2 \cdot POP(I)}{TOTPOP}$$

$$\sigma_y^2 = \frac{\sum_{I=1}^{NTRCTS} Y(I)^2 \cdot POP(I)}{TOTPOP}$$

The resulting statistics are printed on the standard output medium.

```
      SUBROUTINE NEWCTY  (IPASS)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 9, 1972
C     READS TRACT DATA FROM INPUT MEDIUM MA WHERE DATA IS IN SHORT
C     FORM WITH ONLY LAT,LONG,POPN FOR EACH TRACT. ASSUMES HEADER CARD
C     HAS CITY NAME AND A NEGATIVE LATITUDE IS AFTER THE LAST TRACT.
C     THE SUBROUTINE ALSO COMPUTES CENTER OF GRAVITY AND NS EW SIGMAS
      COMMON/CITYPR/NAMEC(20),BLA(2),TOTPOP,BLB,FLATC,FLONC,FLFCT,
     1BLC(3),SGTX,SGTY,BLD(21),NTRCTS
      COMMON/STA4TA/X(4000),Y(4000),POP(4000),V(4000)
      COMMON/TMPAND/JRAD,NNEND,IPNCHA,IPUNCH,JPKTP,ADJSTF,LSTAPE,LSTC
     1 ,NSP,DESMX,FMAXWP
      COMMON/IOPR/BLE,MQ,BLF,MP,BLG(14)
      XSQ = 0.
      YSQ=0.
      FLATC = 0.
      FLONGC = 0.
      TOTPOP = 0.
      READ(MP,3)(NAMEC(I),I = 1,20)
    3 FORMAT(20A4)
C     IFITHIS IS THE LAST CITY AND CITY NAME IS NNEND RETURN WITH
C     IPASS = 1. OTHERWISE IPASS = 0
      IPASS = 0
      IF( NAMEC(1) - NNEND)803,300,803
  300 CONTINUE
      IPASS = 1
      RETURN
  803 CONTINUE
      I = 1
   11 CONTINUE
      READ (MP,4)  Y(I),X(I), POP(I)
    4 FORMAT( 2F10.5,F10.0)
C     TERMINATE CITY READ BY NEGATIVE LATITUDE
      IF(Y(I)) 6,5,5
    5 CONTINUE
      FLATC = FLATC + Y(I)*POP(I)
      FLONGC = FLONGC + X(I) * POP(I)
      TOTPOP = TOTPOP + POP(I)
      I = I + 1
      GO TO 11
C     READING COMPLETED NOW COMPUTE STATISTICS AND FILL ARRAYS
    6 CONTINUE
      NTRCTS = I - 1
      FLATC = FLATC/TOTPOP
      FLONGC = FLONGC/TOTPOP
      FLFCT=COS (FLATC*3.14159/180.)
      DO 150 I=1,NTRCTS
      V(I)=POP(I)
      X(I) = (-X(I) + FLONGC)*60.*FLFCT
```


PRECEDING PAGE BLANK-NOT FILMED

```
      Y(I)=(Y(I)-FLATC)*60.
      XSQ=XSQ+X(I)*X(I)*POP(I)
      YSQ=YSQ+Y(I)*Y(I)*POP(I)
  150 CONTINUE
      XSQ = XSQ /TOTPOP
      YSQ = YSQ/TOTPOP
      SGTX=SQRT (XSQ)
      SGTY=SQRT (YSQ)
C     OUTPUT TRACT STATISTICS ON STANDARD OUTPUT MEDIUM
      WRITE(MQ,27)
   27 FORMAT( 1H1, ////////)
      WRITE(MQ,21)(NAMEC(I),I = 1,15)
   21 FORMAT(1H0, 34HNEW TRACT DATA FOR THE CITY NAMED    ,15A4)
      WRITE(MQ,22) FLATC,FLONGC,TOTPOP,NTRCTS
   22 FORMAT(1H  ,12HLATITUDE IS ,F8.5,4X,13HLONGITUDE IS  , F9.5, ,4X,
     1  20HTOTAL POPULATION IS ,F9.0,4X,13HNO. TRACTS IS  ,I5)
      WRITE (MQ,23) SGTX,SGTY
   23 FORMAT( 1H  ,32HSTD. DEV. IN E - W DIRECTION IS ,F9.5, 4X,
     132HSTD. DEV. IN N -S DIRECTION IS  ,F9.5 )
      RETURN
      END
```

SUBROUTINE OPTWPN (DECK #3)

A. GENERAL

This subroutine optimizes the laydown of weapons in a city to maximize blast kill. The optimization method is taken from the program DBZSEL, originally DEVELOPED ~~describe~~d at IDA by H. Everett in 1964. This program has seen extensive use with detailed analysis of results.

One input to the subroutine is a set of points with values associated with each point. The primary use has been to have these points represent census tracts. The subroutine is also given probability of kill vs. distance curves. Weapons are sequentially optimized in position, with each weapon being located to maximize expected kill. After each weapon is located, the expected loss in value is subtracted from each point. Thus each weapon is optimized against the expected surviving value from all previous weapons.

The location of each weapon is found by evaluating the payoff to the weapon at each of a set of grid points, and choosing that point which maximizes payoff. A fin~~d~~er grid is searched in the vicinity of the located point to refine the optimum location. If A local minima had been missed SOME ~~a~~ later weapon would have larger payoff than an earlier one. In such a case the string of earlier weapons with smaller payoff would be removed and the search be done from the earlier point using the just located weapon with a larger payoff.

B. REQUIREMENTS ON CALLING PROGRAM

The calling program is required to provide locations and values at each tract point in the common block /ST~~YY~~44TA/. The arrays of kill probability, distance squared and first differences, i.e., PKL, DSQL, DPKL, and DDSQL must be provided in the common block /PKPR/ as well as the number of entries in the array, NDS, and the maximum blast screening distances, DMAX. In the array /CITYPR/ the city name, center of gravity and standard deviation of population, as well as the total population and number of tracts are also needed. In the common block /WPNPR/ the weapon reliability must be defined.

In the common block /TMPAND/ the values of DESMX and MAXWP determine the number of weapons to be allocated. The optimization is stopped when either the payoff for the last weapon falls below DESMX or the number of weapons allocated becomes more than FMAXWP. The parameters IPUNCH controls final printout. An absolute value of IPUNCH not equal to 1 suppresses punching, while a negative value suppresses printing results.

The subroutine has parameters NMSH and NWOPT ~~with~~ IN THE calling sequence NMESH is 1/2 the number of mesh lines in each direction. Setting NMESH equal to 0 gives a grid spacing ABOUT equal to the WEAPON lethal radius. Setting NWOPT equal to 1 lists ON THE OUTPUT MEDIUM the location for each weapon and the optimization proceeds. ~~and~~ THIS allows observing how many weapons are removed by the optimization process.

The subroutine returns values of weapon locati~~ng~~ONS, of the array V ~~X~~ and payoff in the common block /WPNPRB/. The values in the

common block ST44TA are the surviving population after the optimization. As mentioned, the subroutine will also punch cards with weapon locations if requested.

C. ALGORITHMS IMPLEMENTED

A quantity DZER is determined ~~as~~ which is close to the mean lethal Radius ~~radius~~ as that distance in the kill array (~~psq~~ DSQ) which yields a kill probability (from the array PK) less than 1/2 the kill probability of the 25th element (near the weapon aim point). The parameter STMIN is a minimum spacing of interest and is set equal to 1/8 DZER.

If the calling parameter of the subroutine NMESH is not zero this value is used as the number of mesh points. If it is than the number of mesh points NMESH is computed as the largest integer less than or equal to

$$\frac{3\sqrt{SGTX \cdot SGTY}^{\,2}}{DZER}$$

where SGTX and SGTY are the city standard deviation of population.

This value of NMSH should offer an acceptable compromise between time spent in mesh searching and time spent in replacing removed weapons.

An initial grid spacing in the x (E-W) and y (N-S) direction is now determined by ~~divider~~ taking the largest integer less than or equal to NMSH3 and dividing SGTX and SGTY by this value. Thus, for example, if NMSH is divisible by three the x grid spacing, xs, is 3 x SGTX/NMSH.

If the grid spacing is less than STMIN it is ~~repeatedly~~ doubled.

For each weapon optimized the following procedure is followed:

The initial aim point is set 6 grid spacings below and 6 grid spacings to the left of the population center of gravity. The aim points are moved across the grid until they are 6 to the right, then each line is searched untill it is 6 above the CG. Thus, the total number of points searched is $(2 \cdot NMSH+1)^2$. This occurs over an area within 3 standard deviations of the population center of gravity.

For each aim point the weapon pay is determined for these tracts with a square of side 2 DMAX centered at the aim point. The square of the distance is computed for these tracts and the kill probability associated with this distance is found by linear interpolation in the array PK and DSQ. This kill probability multiplied by weapon delivery probability, DEL, and the current tract value to obtain an expected value destroyed. This is summed over all tracts to obtain a value destroyed (PAY) at each grid aim point. The maximum of this aim point is found.

For the aim point the grid spacings xs and ys are divided by three and a 5x5 grid centered on the aim point is searched, and a new maximum is found in this finer grid. If the finer grid spacing is larger than the minimum value STMIN the process is repeated with finer grids until a small enough grid has been searched.

The payoff from the current weapon is compared with the payoff of the previous weapon. If it is greater the previous weapon is removed from the weapon list. The value is restored by computing

the survival probability at each point and dividing the value by the survival probability. The next previous weapon payoff is again compared and all weapons are removed until one with greater payoff than the previous weapon is found. This procedure insures that local minima are not missed in the searching process.

The current weapon is added to the weapon list. The value remaining for each tract is reduced by computing the kill probability, P, for each tract and multiplying V by 1-P for the new value of V.

If the parameter NWOPT equals 1 the weapon location and pay just found are listed.

The process of optimizing weapons is continued until either the weapon pay computed falls below DESMX or the number of weapons equals FMAXWP. After this occurs, cards with each weapon location are punched if /IPUNCH/ is one, and weapon locations are listed if IPUNCH is positive.

```
      SUBROUTINE OPTWPN(NMESH,NWOPT)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 15, 1972
C     GIVEN VALUE TRACTS FOR A CITY AND KILL PROBABILITY TABLE FOR
C     WEAPONS, THIS SUBROUTINE OPTIMIZES THE WEAPON LAYDOWN.
C     THIS SUBROUTINE IS BASED UPON THE PROGRAM OGZSEL DEVELOPED BY
C     H. EVERETT III AND LONG USED BY IDA AND WSEG.
C     NMESH IS THE NUMBER OF MESH POINTS. NWOPT = 1 LISTS EACH WEAPON
C     AS LAID DOWN.

      COMMON/TMPANO/JRAD,NNEND,IPNCHA,IPUNCH,JPKTP,ADJSTF,LSTAPE,LSTC
     1 ,NSP,OESMX,FMAXWP
      COMMON/PKPR/NDS,PK(30),DELPS(30),DSQ(30),DELDS(30),BLA(120),DMAX
      COMMON/ST44TA/X(4000),Y(4000),POP(4000),V(4000)
      COMMON/WPNPRB/IWP,XZ(150),YZ(150),PAYZ(150),PAYZI(150)
      COMMON/CITYPR/NAMEC(20),BLD(2),TOTPOP,BLE,FLATC,FLONGC,FLFCT,
     1 BLF(3),SGTX,SGTY,BLG(21),NTRCTS
      COMMON/WPNPR/BLM(2), DEL,BLN(23)
      COMMON/IOPR/BLB,MQ,MS,BLC(15)

      MAXWP = FMAXWP
      IPUNCA = ABS(IPUNCH)
      NMSH = NMESH
      IF( NWOPT .NE. 1) GO TO 41
      WRITE(MQ, 42) (NAMEC(I), I = 1,20)
   42 FORMAT(1H1,///,1H0,10X, 60HWEAPON BY WEAPON LAYDOWN IN BLAST OPTIM
     1IZATION FOR CITY OF      ,/,30X, 20A4,////)
      WRITE(MQ, 43)
   43 FORMAT(1H0, 5H NO.  ,     15H TOTAL VALUE    , 15HVALUE THIS WPN.
     1    , 15H WPN. LONGITUDE   , 15H WPN. LATITUDE    ,/ )
   41 CONTINUE

C     FIND SMALL DISTANCE
      PKTAR = PK(25)/2.
      DO 21 J = 1,NDS
      JJ = NDS - J + 1
      IF(PK(JJ) .GE.PKTAR) GO TO 21
      JUSE = JJ
      GO TO 22
   21 CONTINUE
      JUSE = 1
   22 CONTINUE
      DZER = SQRT(DSQ(JUSE))
      STMIN = DZER/8.

C     ADJUST NUMBER OF MESHES TO GIVE MESH SPACING ABOUT EQUAL TO
C     WEAPON RADIUS.
      IF(NMSH .NE. 0) GO TO 23
      TEMP = SQRT(SGTX*SGTY)/DZER
      NMSH = 3.*TEMP
   23 CONTINUE
```


PRECEDING PAGE BLANK-NOT FILMED

```
C       SETUP INITIAL MESH
        TMP = NMSH/3
  194   XS = SGTX/TMP
        YS = SGTY/TMP
        XC=0.
        YC=0.
        NX = NMSH
        NY = NMSH
        IF(XS-STMIN)360,360,362
    360 IF(YS-STMIN)361,361,362
    361 XS=2.*XS
        YS=2.*YS
        NX = NX/2
        NY = NY /2


        IWP=1
        PAYTOT=0.

    362 CONTINUE
        PAYMAX=0.
C       EVALUATE GRID XC,XS,NX,YC,YS,NY
C       AND SAVE BEST IF BETTER PAYMAX IN XMAX,YMAX
    180 FNY=NY
        FNX=NX
        NNX=2*NX+1
        XAIM=XC-(FNX+1.)*XS
        DO 177 KX=1,NNX
        XAIM=XAIM+XS
        NNY=2*NY+1
        YAIM=YC-(FNY+1.)*YS
        DO 177 KY=1,NNY
        YAIM=YAIM+YS
        JSW=1
        ASSIGN 175 TO NEXT
        GO TO 170
    175 IF(PAY-PAYMAX)  177,177,176
    176 PAYMAX=PAY
        XMAX=XAIM
        YMAX=YAIM
    177 CONTINUE
        IF(XS-STMIN)  178,178,179
    178 IF(YS-STMIN)  190,190,179
    179 XS=XS/3.
        YS=YS/3.
        NX=2
        NY=2
        XC=XMAX
        YC=YMAX
        GO TO 180
    190 PAYZ(IWP)=PAYMAX
        XZ(IWP)=XMAX
        YZ(IWP)=YMAX
        IF(IWP-1)412,412,409
```

```
      409 IF(PAYZ(IWP)-PAYZ(IWP-1))412,412,410
      410 IWP=IWP-1
    C     REMOVE IWP WEAPON
          XAIM=XZ(IWP)
          YAIM=YZ(IWP)
          JSW=3
          ASSIGN 411 TO NEXT
          GO TO 170
      411 PAYTOT=PAYTOT-PAYR
          GO TO 190
      412 CONTINUE
    C     UPDATE
          XAIM=XMAX
          YAIM=YMAX
          ASSIGN 191 TO NEXT
          JSW=2
          GO TO 170
      191 PAYTOT=PAYTOT+PAY
          PAYZ(IWP)=PAY
          PAYZT(IWP) = TOTPOP - PAYTOT

          IF( NWOPT .NE. 1) GO TO 168
    C     PRINT VALUES FOR WEAPON JUST FOUND
          XZP = XZ(IWP)/(FLFCT*60.) + FLONGC
          YZP = YZ(IWP) /60. + FLATC
          WRITE(MQ,203) IWP, PAYZT(IWP),PAYZ(IWP),XZP,YZP
    203   FORMAT(I5, 4F15.6)
    168   CONTINUE
          IF(PAY-DESMX)200,200,192
      192 IF(IWP-MAXWP)  193,200,200
      193 IWP=IWP+1
          GO TO 194
    200   CONTINUE


    C     WRITE FINAL RESULTS OF WEAPON OPTIMIZATION
          IF( IPUNCA .EQ. 1 .OR. IPUNCH  .GT. 0) GO TO 45
          RETURN
    45    CONTINUE
          IF(IPUNCA .NE. 1) GO TO 46
          WRITE(MS,202)(NAMEC(I),I=1,2)    ,TOTPOP,FLATC,FLONGC,IWP,NTRCTS
    202   FORMAT( 2A4,2X,3F15.6,I5, I8)
    46    CONTINUE
          IF( IPUNCH.LT. 0) GO TO 47
          WRITE(MQ,51) (NAMEC(I),I = 1,20),TOTPOP,FLATC,FLONGC, SGTX,SGTY,
         1 NTRCTS,IWP
    51    FORMAT( 1H1 , //////    , 20X, 37HRESULTS OF WEAPON BLAST OPTIMIZAT
         1ION    ,///, 10X,  9H CITY OF    , 20A4, //,  18HCITY POPULATION =,
         2 18H LATITUDE OF CG = , F11.5, 19H LONGITUDE OF CG = ,F11.5,/,
         3 22H E-W POPN STD. DEV. =  ,F10.3,22H N-S POPN STD. DEV. = ,F10.3,
         4  17H NO. OF TRACTS =  ,I6,///, 20X,  I5,18H WEAPONS WERE USED,//)
          WRITE(MQ, 43)
    47    CONTINUE
          DO 52 J = 1,IWP
          XZP =-XZ(1)/(FLFCT*60.) + FLONGC
```

```
      YZP = YZ(I)/60. + FLATC
      IF(IPUNCA .NE. 1) GO TO 53
      WRITE(MS,55) J, PAYZT(J),     PAYZ(J), XZP,YZP
   55 FORMAT(        I5,4F15.6)
   53 CONTINUE
      IF(IPUNCH .LT. 0) GO TO 54
      WRITE(MQ,56) J, PAYZT(J),     PAYZ(J), XZP,YZP
   56 FORMAT( 1H  , I5,4F15.6)
   54 CONTINUE
   52 CONTINUE
      RETURN


C     EVALUATE OR UPDATE AS JSW=1, OR 2
C     REMOVE IF JSW IS 3 RECOVERED PAY TO PAYR
C     EXIT TO NEXT
  170 PAY=0.
      PAYR=0.
      DO 161 I=1,NTRCTS
      TEMPX = ABS(XAIM - X(I))
      IF( TEMPX.GT. DMAX) GO TO 161
      TEMPY = ABS(YAIM - Y(I))
      IF(TEMPY .GT.DMAX) GO TO 161
      DD = TEMPX*TEMPX + TEMPY*TEMPY
      J = 1
  164 CONTINUE
      J = J + 1
      IF(DSQ(J) .GT. DD) GO TO 164
      P =  PK(J) +(DD - DSQ(J))*DELPS(J)/DELDS(J)
      P = P*DEL
      PAY=PAY+V(I)*P
      GO TO (161,160,400),JSW
  400 VPR=V(I)/(1.-P)
      PAYR=PAYR+VPR-V(I)
      V(I)=VPR
      GO TO 161
  160 V(I)=V(I)*(1.-P)
  161 CONTINUE
      GO TO NEXT,( 175,411,191)
      END
```

SUBROUTINE ONEPAS (DECK #4)

A. GENERAL

This subroutine alters the values associated with a number of tracts based upon the input aim point of a single weapon. It is the same as a portion of the subroutine OPTWPN, but allows the user flexibility in location of weapons.

The subroutine calling parameters are the X(E-W) and Y(N-S) coordinates of the weapon to be evaluated and a parameter JSW. The various values of JSW do the following:

JSW =	1	only determines weapon payoff
	2	determine weapon payoff and removes value
	3	determine weapon payoff and replaces population as if weapon is removed

The calculated weapon payoff is returned in the parameter PAY in the calling sequence.

B. REQUIREMENTS ON CALLING PROGRAM

Besides the parameters in the subroutine calling sequence, the calling program must supply the following block ~~computer~~ COMMON variables (AS DEFINED IN SUBROUTINE OPTWPN):

in /PKPR/ - PK, DELPS, DSQ, DELDS, DMAX

/STYYTA/ - X, Y, V

/CITYPR/ - NTRCTS

/WPNPR/ - DLL

~~The definition of this variable is the same as in subroutine OPTWPN.~~ For values of JSW = 2, 3 the values in V are updated.

C. ALGORITHMS IMPLEMENTED

Calculations of probability of kill are made for each tract in a square centered at the weapon input aim point XAIM, YAIM. The probability of kill is obtained by linear interpolation on the distance squared from the aim point to the tract location vs. probability of hit PK. The pay is calculated as the kill probability, P, times weapon delivery probability, DEL, times the tract value, V, and summed for JSW equals 1 or 2. For JSW = 2 the value V is reduced to

$$V(1-P \cdot DEL).$$

For JSW = 3 the new tract value is $V/(1-P \cdot DEL)$. The value of PAY is the summation of new tract values minus old tract values.

```
      SUBROUTINE ONEPAS(JSW,XAIM,YAIM,PAY)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 15, 1972
C     DOES A SINGLE UPDATE TRACT BY TRACT FOR BLAST KILL BY ADDING OR
C     REMOVING A WEAPON
      COMMON/PKPR/BLK,PK(30),DELPS(30),DSQ(30),DELDS(30),BLA(120),DMAX
      COMMON/STDATA/X(4000),Y(4000),POP(4000),V(4000)
      COMMON/CITYPR/ BLD(55),NTRCTS
      COMMON/WPNPR/BLM(2), DEL,BLN(23)
C     EVALUATE OR UPDATE AS JSW=1, OR 2
C     REMOVE IF JSW IS 3 RECOVERED PAY TO PAYR
  170 PAY=0.
      PAYR=0.
      DO 161 I=1,NTRCTS
      TEMPX = ABS(XAIM - X(I))
      IF( TEMPX.GT. DMAX) GO TO 161
      TEMPY = ABS(YAIM - Y(I))
      IF(TEMPY .GT.DMAX) GO TO 161
      DD = TEMPX*TEMPX + TEMPY*TEMPY
      J = 1
  164 CONTINUE
      J = J + 1
      IF(DSQ(J) .GT. DD) GO TO 164
      P =  PK(J) +(DD - DSQ(J))*DELPS(J)/DELDS(J)
      P = P*DEL
      PAY=PAY+V(I)*P
      GO TO (161,160,400),JSW
  400 VPR=V(I)/(1.-P)
      PAYR=PAYR+VPR-V(I)
      V(I)=VPR
      GO TO 161
  160 V(I)=V(I)*(1.-P)
  161 CONTINUE
      IF(JSW.EQ.3) PAY = PAYR
      RETURN
      END
```

SUBROUTINE FLPKA (DECK #5)

A. GENERAL

This subroutine computes a table of probability of kill or injury as a function of distance. It assumes that the probability is a cumulative normal function of the logarithm of the overpressure. The overpressure is a function of distance computed on the basis of weapon yield and height of burst by the subroutines PROMPT and PDIST. The effects of CEP are determined by numerically integrating the kill probability times the likelihood of weapon impact over a grid. To normalize the kill function the kill probability is integrated over an area. This area is compared to the area of a circle which has as its radius the distance at which the mean lethal overpressure occurs. The parameter ADJSTF multiplies the differences in distance SQUARED and adds this to the original value. Thus a value of zero leaves the distance unchanged, a value of 1 gives distances which normalize the kill function area, and intermediate values yield intermediate distances.

This subroutine produces values directly of kill probability which are based upon the physical observations of damage in a fashion which allows the user the maximum control in constructing a kill probability damage curve. Its use is preferred to a fit function unless many kill probability tables are to be constructed so that rapidity in calculation becomes an important consideration.

The final output are arrays of kill or injury probability and the square of the corresponding distance. Arrays of first differences are also ~~constructed~~ GIVEN for rapidity in interpolation IN LATER USES.

B. REQUIREMENTS ON CALLING PROGRAM

This subroutine uses common blocks TMPAND, VULPR, PKPR, EFFCAL, and IOPR.

The calling program must supply values for the following control parameters in TMPAND:

IPNCHA	If one - set of 26 cards with probability-distance tables is punched
JPKTP	If two - set of cards with PK-distance tables is read and the subroutine is exited
ADJSTF	Normalization factor; its value must be defined

The following parameters must be defined in VULPR

PSIL	mean lethal overpressure/psi
SGPSIL	Std.Dev. on PSIL for distribution of probability with log of pressure
PSINJ	Mean injury overpressure
SGPINJ	Std. Dev. on PSINJ

When the subroutine is called the following must be defined in WPNPR

CEPW	Weapon CEP
CRTYLW	Cube root of weapon yield
NTYPEW	Weapon burst type identifier for use in PROMPT 0 - surface burst 1 - 10 psi opt. air burst

The subroutines PROMPT and PDIST and the function CUMNOR are needed by this subroutine.

C. ALGORITHMS IMPLEMENTED

If the parameter TPRTP is 2, cards are read with tabular values and the subroutine is exited.

The array PKT contains the following 26 entries for probability:

.0001, .001, .01, .02, .05, .10, .15, ..., .90, .95, .98, .99, .9999 .

The array RLGPP contains the values of x in a cumulative normal distribution FOR THE ABOVE PROBABILITY VALUES. For both lethal and injury psi the log of the pressure is calculated by

$$\log_{10} p = (\text{RLGPP} \cdot \text{SGPSIL} + 1) \log_{10}(\text{psi})$$

where

psi is the overpressure for 50% injury

SGPSIL is the standard deviation of the cumulative normal distribution

The distance at which this pressure occurs is calculated by the routine PDIST. (For the 10 psi airburst the distance is set equal to zero if the pressure is over 30 psi.) A table of distance versus probability is printed.

The distance for the first probability entry (p = .0001) is set equal to a very large value and for the 26th (p = .9999) equal to 0. The distance for the 24th entry (p = .98) and 25th entry (p = .99) is set equal to 1/3 and 2/3 of the distance for the 23rd entry (p = .95). For the third entry, this distance is increased by 4 TIMES the aiming error, SIG, for the weapon (CEP/1.1774). These adjustments are made to yield an interpolation table with better spacing for the distance arguments. Since the probabilities are about to be recalculated, no error is introduced during this.

Next, a mesh of 20 x 20 points is constructed centered at the desired ground zero with a spacing of SIG /2.5. For each grid point the distance from the grid point to the target is computed.

The pressure at this grid point is computed by the subroutine PROMPT, and the kill probability is computed as the cumulative normal function of pressure,

$$\text{Prob.cumnor}((\log_{10}\text{pressure}-\log_{10}\ \text{PSIL})/\text{SGPSIL})\ .$$

This probability is multiplied by the probability of weapon impact, assuming weapon delivery errors are circularly normally distributed. Simpson's rule is then used in two dimensions to calculate the integral value. Finally, the probability values for the final two entries are set equal to zero because of the large distance assumed for the first distance.

The area integral of the probability values is computed. The ratio of this area to π times the square of the distance for the mean overpressure is calculated for both the lethal and injury curves. The square of the distances are increased by the ratio minus 1 times the input parameter ADJSTF. Normalization values and adjusted differences are printed out.

First differences of probabilities and square of the distance are calculated for assistance in rapid interpolation between array values. The maximum distance for which the kill probability is not zero is given as DMAX.

If the parameter IPNCHA is one of the probabilities, values and distances are punched on cards.

```
      SUBROUTINE FLPKA
C     NEVUNS STANDARD
C     LAST REVISED ON NOV, 6, 1972
C     FILLS THE ARRAYS DSQI,PKCI,DSQK,PKCK,DELDSI,DELPSI,DELDSK,DELPSK
C     WITH PRESSURE PK DISTANCE RELATIONSHIPS DIRECTLY CALCULATED
C     THE EFFECTS OF CEP ARE INCLUDED BY DIRECT INTEGRATION OF PROBS
      DIMENSION PKT(26) , RLGPP(26),DBL(26),DHC(26)
      COMMON/TMPAND/JRAD,NNEND,IPNCHA,IPUNCH,JPKTP,ADJSTF,LSTAPE,LSTC
     1 ,NSP,DESMX,FMAXWP
      COMMON/PKPP/NDS,PKCK(30),DELPSK(30), DSQK(30),DELDSK(30),
     1 PKCI(30),DELPSI(30), DSQI(30), DELDSI(30),DMAX
      COMMON/VULPR/PSI,SIGBL,PSINJ,SIGHC,BLANK(103)
      COMMON/WPNPR/BLA(3),CEP,BLB(6),NTYPE ,BLC(13),YLUNU,BLD
      COMMON/EFFCAL/BLE,YLDNO,BLF,JTINR,JHTPR,DSTP,BLG,PRESS,PRELP,
     1 BLH(10)
      COMMON/IOPR/MP,MQ,MS,BNA(15)
      DATA PKT/.0001,.001,.01,.02,.05,.10,.15,.20,.25,.30,.35,.40,.45,
     1 .50,.55,.60,.65,.70,.75,.80,.85,.90,.95,.98,.99,.9999/
      DATA RLGPP/-4.4170,-3.2905,-2.32635,-2.05375,-1.64475,-1.28155,
     1 -1.03643,-0.84162,-0.67449,-0.52440,-0.38532,-0.25335,-0.12566,
     2 0.0,0.12566,0.25335,0.38532,0.52440,0.67449,0.84162,1.03643,
     3 1.28155,1.64475,2.05375,2.32635,4.4170/

      IF(JPKTP .NE. 2) GO TO 1
C     READ DATA INTO ARRAYS AND DO NOT CALCULATE IT.
      DO 5 K = 1,26
      READ(MP   ,52)J,DSQI(J),PKCI(J)   ,DSQK(J),PKCK(J)  ,J,DELDSI(J),
     1DELPSI(J),DELDSK(J) ,DELPSK(J)
    5 CONTINUE
      RETURN
    1 CONTINUE

C     TO USE IN /EFFCAL/
      YLDNU = YLDNO
      JHTPR = NTYPE
      JTINR = 0
C     FIRST FIND VALUES OF PRESSURE AND THEN DISTANCE FOR THAT PRESS.
      RLPM = ALOG10(PSI)
      RCPM = ALOG10(PSINJ)
      SIGPL = RLPM*SIGBL
      SIGPC = RCPM*SIGHC
      DO 10 J = 1,26
      PLL = RLGPP(J)*SIGBL*RLPM + RLPM
      IF( NTYPE .NE. 0) GO TO 7
      IF( PLL .GT. 1.57978) GO TO 6
      GO TO 7
    6 DIST = 0.
      GO TO 8
```

```
    7     CONTINUE
          CALL PDIST(NTYPE,1,RLL,DIST)
          DIST = DIST*YLONU
    8     CONTINUE
          DBL(J) = DIST
          DSQK(J) = DIST*DIST
          RCL = RLGPP(J) *SIGBC* RCPM + RCPM
          IF (NTYPE .NE. 0) GO TO 3
          IF( RCL.GT. 1.57978)  GO TO 2
          GO TO 3
    2     DIST = 0.
          GO TO 4
    3     CONTINUE
          CALL PDIST(NTYPE,1,RCL,DIST)
          DIST = DIST*YLONU
    4     CONTINUE
          DBC(J) = DIST
          DSQI(J) = DIST*DIST
   10     CONTINUE
    C     OUTPUT INITIAL CALCULATIONS.
          WRITE(MQ,15)
   15     FORMAT(1H1)
          WRITE(MQ,16)
   16     FORMAT(///////)
          WRITE (MQ,11)
   11     FORMAT( 1H0, 36HZERO CEP PK DIST LETHAL AND INJURY     )
          WRITE (MQ,12)
   12     FORMAT( 1H0, 4H NO., 8H  PROB    ,2X, 8H DIST L  ,4X,8H DIST I
         1,12X  , 4H NO.  , 8H  PROB    ,2X,   8H DIST L  ,4X, 8H DIST I    )
          DO 14 I = 1,13
          II = 2*I
          IM = II - 1
          WRITE(MQ,13) IM,PKT(IM) , DBL(IM)  ,DBC(IM) ,II,PKT(II),
         1DBL(II),,DBC(II)
   13     FORMAT(1H  ,1H(, I2,1H), F8.6,2F12.6,10X,1H(,I2,1H),
         1F8.6,    2F12.6)
   14     CONTINUE
    C     ADJUST DISTANCES TO GET A BETTER INTERPOLATION TABLE.
          DSQK(1) = 0.79012344E+10
          DBL(1) = 8888.8888
          DSQI(1) = 0.79012344E+10
          DBC(1) = 8888.8888
          DBL(24) = 0.666667*DBL(23)
          DBL(25) = 0.333333*DBL(23)
          DBC(24) = 0.666667*DBC(23)
          DBC(25) = 0.333333*DBC(23)
          DSQK(24) = DBL(24)*DBL(24)
          DSQK(25) = DBL(25)*DBL(25)
          DSQI(24) = DBC(24)*DBC(24)
          DSQI(25) = DBC(25)*DBC(25)
          DBL(26) = 0.
          DBC(26) = 0.
          DSQK(26) = 0.
```

```
        DSQI(26) = 0.
        SIG = CEP/1.1774
        DBL(2) = DBL(2) + 4.*SIG
        DBC(2) = DBC(2) + 4.*SIG
        DSQK(2) = DBL(2)*DBL(2)
        DSQI(2) = DBC(2)*DBC(2)
C       SETUP FOR NUMERICAL INTEGRATION FOR CEP EFFECTS.
        DIL = SIG/2.5
        SIGS = SIG*SIG
        TPSIGS =1./(2.*3.14159265*SIGS)
        TSIGS = 1./(2.*SIGS)
C       INTEGRATE FOR EACH DISTANCE
        DO 100 JK = 3,26
        SUMCL = 0.
        SUMCC = 0.
        SUMCP = 0.
        CROSS = -DIL
C       CTRC AND CTRR USED AS WEIGHTS IN SIMPSONS RULE
        CTRC = 1.
        DO 20 K = 1,21
        CROSS = CROSS + DIL
        CTRR = 1.
        DOWN = -11.*DIL
        SUMRL = 0.
        SUMRC =  0.
        SUMRP = 0.
        DO  30 J = 1,21
        DOWN = DOWN + DIL
        RAD = CROSS*CROSS + DOWN*DOWN
        PRORD = TPSIGS*EXP(-RAD*TSIGS)
        T1 = DBL(JK) - DOWN
        T2 = T1*T1 + CROSS*CROSS
        DSTP = SQRT(T2)
        CALL PROMPT
        XLPL = PRELP
        T1 = DBC(JK) - DOWN
        T2 = T1*T1 + CROSS*CROSS
        DSTP = SQRT(T2)
        CALL PROMPT
        XLPC = PRELP
        T1 =  ( XLPL - RLPM)/SIGPL
        PROL = CUMNOR(T1)
        T2 =  (XLPC - RCPM)/SIGPC
        PROC = CUMNOR(T2)
        IF( J.EQ. 21) CTRR = 1.
        SUMRL = SUMRL + CTRR*PRORD*PROL
        SUMRC = SUMRC + CTRR*PRORD*PROC
        SUMRP = SUMRP + CTRR*PRORD
        IF( CTRR - 2.) 31,31,32
31      CONTINUE
        CTRR = 4.
```

```
              GO TO 33
   32         CONTINUE
              CTRR = 2.
   33         CONTINUE
   30         CONTINUE
              IF( K.EQ. 21 ) CTRC = 1.
              SUMCL = SUMCL + CTRC*SUMRL
              SUMCC = SUMCC + CTRC*SUMRC
              SUMCP = SUMCP + CTRC*SUMRP
              IF( CTRC - 2.) 21,21,22
   21         CONTINUE
              CTRC = 4.
              GO TO 23
   22         CONTINUE
              CTRC = 2.
   23         CONTINUE
   20         CONTINUE
   C          PDELN IS USED TO NORMALIZE PROBABILITY INTEGRAL SINCE INTEGRATION
   C          IS NOT EXACT
              PDELN = 2.*SUMCP*DIL*DIL/9.
              PKCK(JK) = 2.*SUMCL *DIL*DIL/(9.*PDELN)
              PKCI( JK) = 2.*SUMCC*DIL*DIL/(9.*PDELN)
  100         CONTINUE
   C          ADJUSTMENTS FOR INTERPOLATION  TABLE SINCE DBL(I) IS LARGE.
              PKCI(I) = 0.
              PKCK(1) = 0.
              PKCI(2) = 0.
              PKCK(2) = 0.
   C... INTEGRATE OVER LETHAL AREA TO NORMALIZE KILL FUNCTIONS
              TPS1 = 3.14159265
              TPS2 = 0.5*3.14159265
              SINTK = 0.
              SINTI = 0.
              DO 70 I = 3,26
              BKI = (PKCK(I) - PKCK(I - 1))/ (DSQK(I) - DSQK(I - 1))
              AKI = PKCK(I -1) - DSQK( I-1)*BKI
              BII = (PKCI(I)- PKCI(I-1) )/(DSQI(I   ) -DSQI(I-1)    )
              AII = PKCI(I-1)-DSQI(I-1)*BII
              SINTK = SINTK -TPSI*AKI*( DSQK(I)        - DSQK(I-1)          )
             1 - TPS2*BKI*( DSQK(I)*DSQK(I)       - DSQK(I-1)*USQK(I-1) )
              SINTI = SINTI -TPSI*AII*(        DSQI(I) - DSQI(I-1)  )
             1-TPS2*BII*( DSQI(I)*DSQI(I)        -DSQI(I-1)*DSQI(I-I)  )
   70         CONTINUE
              AKN = 3.14159265*DSQK(14)
              AIN = 3.14159265*DSQI(14)
              RAK = AKN/SINTK
              SRAK = SQRT(RAK)
              RAI = AIN/SINTI
              SRAI = SQRT(RAI)
              WRITE (MO,71) SRAK,SRAI,ADJSTF
   71         FORMAT(1H0,53HRATIO OF LETHAL RADIUS AREA TO CEP INTEGRATED AREA 1
             1S   ,F10.5 ,21H FOR FATALITIES, AND ,F10.5,13H FOR INJURIES,
```

```
      2 /,27H DISTANCES ARE ADJUSTED BY  ,F5.3, 14H OF THIS RATIO
      3, 13H TO NORMALIZE  )
       DO 72 J = 1,26
       DSQK(J) = DSQK(J)*( 1. +(RAK-1.)*ADJSTF)
       DBL(J) = DBL(J)*( 1. + (SRAK-1.)*ADJSTF)
       DSQI(J) = DSQI(J)*( 1. + (RAI -1.)*ADJSTF)
       DBC(J) = DBC(J)*( 1.+ (SRAI - 1.)*ADJSTF)
 72    CONTINUE
       WRITE(MO,17)
 17    FORMAT(//)
       WRITE( MO,44)
 44    FORMAT( 1H0, 42HCEP INTEGRATED PK DIST LETHAL AND INJURY         )
       WRITE (MO,43)
 43    FORMAT( 1H0,  4H NO. ,9H  PROB L  ,2X, 8H DIST L  ,2X,
      19H  PROB I ,2X, 8H DIST I   , 12X,4H NO.   ,9H  PROB L      ,
      2 2X, 8H DIST L ,2X, 9H  PROB I  ,2X, 8H DIST I  )
       DO 41 I = 1,13
       II = 2*I
       IM = II - 1
       WRITE (MO, 42) IM,PKCK(IM), DBL(IM), PKCI(IM), DBC(IM),
      1II,PKCK(II),DBL(II), PKCI(II), DBC(II)
 42    FORMAT( 1H  , 1H(,I2,1H), F9.6, F12.6,F9.6, F12.6,10X,
      11H(,I2, 1H), F9.6,F12.6, F9.6, F12.6     )
 41    CONTINUE

C      FILL DIFFERENCE TABLE
       DO 61 I = 2,26
       DELDSI(I) = DSQI(I) - DSQI(I - 1)
       DELPSI(I) =  PKCI(I) - PKCI(I - 1)
       DELDSK(I) = DSQK(I ) - DSQK(I - 1)
       DELPSK(I) = PKCK(I) - PKCK(I - 1)
 61    CONTINUE
       DMAX = DBL(2)

       IF (IPNCHA .NE. 1) GO TO 51
C      PUNCHED CARD OUTPUT
       DO 53 J = 1,26
       WRITE( MS,52)J,DSQI(J),PKCI(J)  ,DSQK(J),PKCK(J) ,J,DELDSI(J),
      1DELPSI(J),DELDSK(J) ,DELPSK(J)
 52    FORMAT(I4,4E15.9,/,  I4,4E15.9 )
 53    CONTINUE
 51    CONTINUE
       RETURN
       END
```

SUBROUTINE FLPKHU (DECK #6)

A. GENERAL

This subroutine fills arrays with probability of kill and injury for future use in optimization routines. The method is based upon use of the SSKP function as outlined in LAMBDA Paper 6 and in the SSKP writeup. The square of the distance at which various values of SSKP is found for both lethal and injury overpressures is found.

This subroutine has been used extensively in many IDA/WSEG calculations, AND SIMILARLY BASED CALCULATIONS HAVE BEEN USED ELSEWHERE. It was used in Program DGZSEL as part of the IDA/WSEG DAL and CANOPY studies. It was used in AGZSEL for IDA Paper P-762, and in Program ANDANTE in blast optimization calculations. Because of this long history the capability is retained in the NEVUNS system to provide a means of correlation with previous results. The subroutine FLPKA is to be preferred if a close relation to the causes of blast damage is desired.

B. REQUIREMENTS ON CALLING PROGRAM

The block common communications are as follows:

To be filled by calling program:

/WPNPR/	CEP	weapon delivery error
	NTYPE	0 for surface burst -1 for optimum air burst
	YLDNU	cube root of yield
/VULPR/	PSIL	median lethal overpressure
	PSINJ	median injury overpressure

XMOD shape parameter in WSEG model not currently needed since MOD is set to 0 ~~be~~ in the program. Readily implemented by changing the first line of coding from MOD = 0 to MOD = XMOD

/IOPR/ standard output medium definition

Filled by subroutine for temporary use

/TRNSFR/

Results available in PKPR

NDS number of entries in table (= 26)

PKL(I) kill probability

DPKL(I) PKL(I)-PKL(I-1)

DSQL(I) square of distances for PKL

DDSQL(I) DSQL(I)-DSQL(I-1)

PKI
DPKI
DSQI
DDSQI

same as for lethal but using injury overpressure

DMAX $\sqrt{DSQL(2)}$ maximum distance with none zero kill probability as for use in blast screening calculations

The following functions are needed for this subroutine:

Functions PZZ, SSKP, G, XLRAD, ROOTF.

C. ALGORITHMS IMPLEMENTED

The median overpressure, PSI, is set equal to the median lethal and median injury overpressure and the following calculations are carried out:

The weapon radius, DZER, is found by obtaining a distance from the function XLRAD multiplied by the cube root of the yield.

The standard deviation of weapon aiming error SIGD is found from the CEP by

SIGD = CEP/1.1774

The kill probability with zero weapon ~~effect~~ OFFSET, PMAX, is found from the SSKP function.

The distance for the first array element is set to a large number and the kill probability to zero. The kill probability for the second element is set equal to zero. This is done for use in interpolation schemes USING THESE TABLES.

Target kill probabilities are found as the following fractions of PMAX:

.001, .01, .02, .05, .10, .15, ..., .90, .95, .98, .99 .

This gives 24 more array elements. For each target kill probability the distance at which the SSKD function would yield this kill probability is found by the root finding function. This is accomplished by finding the root of the function, PZZ, which is simply equal to the difference between the SSKP function and the desired kill probability. The last array element sets the distance equal to zero and kill probability equal to one, again for interpolation purposes.

For the third to 25th elements the SSKP function is then entered to compute the kill probability at the computed distance. From the two arrays of distance and kill probability the desired output arrays are computed, i.e., kill probability, distance squared, first difference of kill probability, and first difference of distance squared.

The appropriate external arrays are filled depending upon whether lethal or injury overpressures have been used. The computed distances and kill probabilities are recorded on the output medium.

```
      SUBROUTINE FLPKHU
C     NEVUNS STANDARD
C     LAST REVISED NOV. 9, 1972
C     FILLS PK TABLE ACCORDING TO EVERETT FASHION IN ORIGINAL DGZSEL
C     TABLES FILLED FOR BOTH BLAST AND INJURY
      DIMENSION PKC(30),DSQ(30),DELDS(30),DELPS(30)
      COMMON/PKPR/ NDS,PKL(30),DPKL(30),DSQL(30),DDSQL(30),
     1 PKI(30), DPKI(30),DSQI(30),DDSQI(30), DMAX
      COMMON/WPNPR/BLA(3),CEP,BLB(6),NTYPE ,BLC(13),YLDNU,BLD
      COMMON/VULPR/ PSIL,BLK,PSINJ,BLL(3),XMDD,BLE(102)
      COMMON/TRNSFR/ MDD,DZER,SIGD,PTG,PMAX,BLF(15)
      COMMON/IOPR/ BLG,MQ,BLH(16)
      EXTERNAL PZZ

      MDD=6

      ZILCH = SSKP(0,0.,0.,0.,0.,0.)
      ISTRT = 1
      PSI = PSIL
      GO TO 24
   23 CONTINUE
      PSI = PSINJ
   24 CONTINUE
      IF( NTYPE .LT.0) PSI = - PSI
      DZER = XLRAD(PSI)*YLDNU
      SIGD=CEP/1.1774
      PMAX = SSKP(MDD, DZER,SIGD,SIGD,0.,0.)
      DDMAX=5.*(DZER+CEP)
      DSQ(1)=1.E50
      PKC(1)=0.
      PTG=.001
      DSQ( 2) = ROOTF(0.,DDMAX,.001,PZZ)
      PKC(2)=0.
      PTG=.01
      DSQ( 3) = ROOTF(0.,DDMAX,.001,PZZ)
      PTG=.02
      DSQ( 4) = ROOTF(0.,DDMAX,.001,PZZ)
      PTG=.05
      DO 700 I=5,23
      DSQ( I) = ROOTF(0.,DDMAX,.001,PZZ)
      PTG=PTG+.05
  700 CONTINUE
      PTG=.98
      DSQ(24) = ROOTF(0.,DDMAX,.001,PZZ)
      PTG=.99
      DSQ(25) = ROOTF(0.,DDMAX,.001,PZZ)
      DSQ(26)=0.
      PKC(26)=1.
      DO 701 I=3,25
      XXXX = DSQ(I)
```

```
            PKC(I) = SSKP(  MOD,DZER,SIGD,SIGD,XXXX,0.)
            DSQ(I)=DSQ(I)*DSQ(I)
      701   CONTINUE
            DSQ(2)=DSQ(2)*DSQ(2)
            NDS=26
            DO 710 I = 2,26
            DELDS(I) = DSQ(I) - DSQ(I-1)
            DELPS(I) = PKC(I) - PKC(I - 1)
     710    CONTINUE

            IF(ISTRT .NE. 1) GO TO 21
            DMAX=SQRT (DSQ(2))
            DZA = DZER
            DO 22 J = 1,NDS
            PKL(J) = PKC(J)
            DPKL(J) = DELPS(J)
            DSQL(J) = DSQ(J)
            DDSQL(J) = DELDS(J)
     22     CONTINUE
            ISTRT = 2
            GO TO 23
     21     CONTINUE
            DZB = DZER
            DO 26 J = 1,NDS
            PKI(J) = PKC(J)
            DPKI(J) = DELPS(J)
            DSQI(J) = DSQ(J)
            DDSQI(J) = DELDS(J)
     26     CONTINUE

            WRITE(MQ, 48)
     48     FORMAT( 1H1,/////)
            WRITE(MQ,8)
     8      FORMAT(1H0, 48HVALUES OF PK - DIST BY DGZSEL TYPE CALCULATION     )
            WRITE (MQ,51) PSIL,DZA,PSINJ,DZB
     51     FORMAT(////,1H0,29HMEDIAN LETHAL OVERPRESSURE =  ,F10.3,
           1 22H WITH LETHAL RADIUS =  ,F10.3,/,10X, 29HMEDIAN INJURY OVERPRES
           2SURE =    ,F10.3,  22H WITH INJURY RADIUS =  ,F10.3 ,//)
            WRITE (MQ,43)
     43     FORMAT( 1H0,  4H NO. ,9H  PROB L  ,2X, 8H DIST L  ,2X,
           19H  PROB I ,2X, 8H DIST I  , 12X,4H NO.   ,9H  PROB L      ,
           2 2X, 8H DIST L ,2X, 9H  PROB I   ,2X, 8H DIST I  )
             DO 41 I = 1,13
             II = 2*I
             IM = II - 1
             TL1 = SQRT(DSQL(IM))
             TL2 = SQRT(DSQL(II))
             TI1 = SQRT(DSQI(IM))
             TI2 = SQRT(DSQI(II))
             WRITE( MQ,42) IM,PKL(IM), TL1,PKI(IM),TI1,
           1 II,PKL(II),TL2, PKI(II),TI2
     42      FORMAT( 1H  , 1H(,I2,1H), F9.6, F12.6,F9.6, F12.6,10X,
           11H(,I2, 1H), F9.6,F12.6, F9.6, F12.6     )
     41      CONTINUE
             RETURN
```

```
      END
```

FUNCTION PZZ (DECK #7)

A. GENERAL

This function is used by the FUNCTION ROOTF to satisfy its formal calling requirements. It is used by FLPKHU to find a value of aiming offset so SSKP has a value of PTG·PMAX.

B. REQUIREMENTS ON CALLING PROGRAM

The following elements must be filled in the common block |TRNSFR|:

MOD
DZER
SIGD
PTG
PMAX

These are now filled by subroutine FLPKHU. This function needs the function SSKP available.

C. ALGORITHM IMPLEMENTED

The function PZZ is defined ~~in~~

PZZ = SSKP(MOD,DZER,SIGN,SIGD,XXXX, 0)-PTG·PMAX .

```
      FUNCTION PZZ(XXXX)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 7, 1972
C     USED IN CONJUNCTION WITH ROOTF TO FILL A SPECIAL ROLE IN FLPKHU
C     THERE IS NO GENERAL USE FOR THIS FUNCTION.
      COMMON/THNSFR/MOD,DZER,SIGD,PTG,PMAX,BLA(15)
      EXTERNAL SSKP
      TEMP = SSKP(MOD,DZER,SIGD,SIGD,XXXX,0)
      PZZ = TEMP - PTG*PMAX
      RETURN
      END
```


PRECEDING PAGE BLANK-NOT FILMED

FUNCTION SSKP (Deck #8)

A. GENERAL

This function is based on work by H. Everett and R. Galiano described in Lambda Corporation Paper No. 6.[1] The coding was originally developed as described in this paper, and has been used in IDA calculations for almost a decade. In particular, a number of previous civil defense calculations have been based on this method.

This function is used to compute the single shot probability of kill from the blast wave with weapon aiming error, and the weapon aimed at a point offset from the intended target. The standard deviation of the aiming error is σ_x, σ_y in the two normal directions and the aim point is offset from the target by a distance μ_x, μ_y.

The probability of kill function is given by

$$G_K(r) = e^{-K} \sum_{j=0}^{W-1} \frac{K^j}{j!}$$

where

$$K = \frac{Wr^2}{a^2}$$

a is the lethal radius

r is the distance from weapon impact to the target

The parameter W allows control of the shape of the function $G_K(r)$. For W = 1 a normal curve is obtained; for W = a cookie cutter curve. According to the Lambda paper: "Standard Kill curves, such as the σ_{20} and σ_{30} curves of AFM 200-8, representing, respectively, ground burst and air burst blast damage probabilities as a function of distance can readily be approximated. W = 6 approximates closely the σ_{20} curve, W = 3 approximates the σ_{30} curve."

1. Robert G. Galiano and Hugh Everett, III, "Some Mathematical Relations for Probability of Kill-Family of Damage Function for Multiple Weapon Attacks", Lambda Corporation, Defense Models IV, March, 1967.

B. REQUIREMENTS ON CALLING PROGRAM

The only communications are the following input values to the function, and the function value which is returned.

The inputs are:

MOD	the value of the shape parameter W in fixed point form
A	the weapon lethal radius
SX(σ_x)	the weapon aiming error in the x direction
SY(σ_y)	the weapon aiming error in the y direction
XMU(μ_x)	the intended weapon offset in the x direction
YMU(μ_y)	the intended weapon offset in the y direction.

It is only required that the distances be in consistent units.

The SSKP function assumes ~~the function~~ certain parameters must be initialized through an initial call on the function with MOD=0. The initialization is accomplished and the function exited with value of SSKP = 0.

The SSKP function uses a special function G which must be available to it.

C. ALGORITHM IMPLEMENTED

The equation implemented is obtained by integrating the function $G_K(r)$ over the elliptical delivery error. The result is equation (15) in Lambda Paper No. 6. It is

$$P_K W = \frac{1}{2\pi\sigma_x\sigma_y} \sum_{j=0}^{W-1} \frac{b^j}{j!} \sum_{\ell=0}^{j} \binom{j}{\ell} H(2\ell,\sigma_x,\mu_x,b) \cdot H(2(j-\ell),\sigma_y,\mu_y,b)$$

where

$$b = \frac{W}{a^2} .$$

The definition of is equation (16) in Lambda Paper No. 6. It is

$$H(M,\sigma,\mu,b) = \left(\frac{\beta}{\alpha}\right)^M \left(\frac{\sigma}{\alpha}\right) \exp\left[-\frac{\beta^2\mu^2}{2\sigma^2}\right] \cdot \sum_{j=0}^{M} \binom{M}{2j} \left(\frac{\sigma}{\beta}\right)^{(j+1)/2} \Gamma\left(\frac{j+1}{2}\right)$$

with

$$\beta_2 = \mu/\alpha$$
$$\alpha = 1+2\sigma^2$$

The equation for H is implemented as the function G.

```
                 FUNCTION SSKP (MOD,A,SX,SY,XMU,YMU)
        C        NEVUNS STANDARD
        C        LAST REVISED NOV. 7, 1972

        C        COMPUTES SINGLE SHOT KILL PROBABILITY FOR ELLIPTICAL NORMAL
        C        DISTRIBUTION WITH AIMING ERROR SX AND SY. THE AIMING OFFSET IS
        C        XMU AND YMU. A IS THE WEAPON LETHAL RADIUS. MOD IS A SHAPE FACTOR
        C        MOD = 1 IS GAUSSIAN IN PK VS ACTUAL DISTANCE FROM THE WEAPON
        C        MOD = 3 IS SIG 20, MOD = 6 IS SIG 30 , MOD = INF IS COOKIE CUTTER
        C        FOR A DESCRIPTION OF EQUATIONS SEE LAMDA PAPER 6 BY HUGH EVERETT
        C         III AND R. GALIANO.
        C        THE FUNCTION MUST BE INITIALIZED BY AN INITIAL CALL WITH MOD = 0
        C        ZERO VALUE OF SSKP IS RETURNED. AFTERWARDS NORMAL CALLS CAN BE
        C        MADE FOR THE DURATION OF THE PROGRAM.

                 DIMENSION BIN(250)
                 DIMENSION W(11)
                 IF(MOD)10,11,10
        11       CONTINUE
        C        INITIALIZATION
                 BIN(1)=1.0
                 BIN(2)=1.0
                 DO 20 J=2,20
                 L=J*(J+1)/2+2
                 BIN(L-1)=1.0
                 BIN(L-2)=1.0
                 NN=J*(J-1)/2+1
                 LAST=J-1
                 DO 20 K=1,LAST
                 BIN(L)=BIN(NN)+BIN(NN+1)
                 L=L+1
                 NN=NN+1
              20 CONTINUE
                 W(1)=2.506628474
                 W(2)=2.506628474
                 W(3)=7.519885422
                 W(4)=37.59942711
                 W(5)=263.1959098
                 W(6)=2368.763908
                 W(7)=26056.40299
                 W(8)=338733.23882
                 W(9)=5080998.582
                 W(10)=86376975.90
                 W(11)=1641162542.
                 SSKP = 0.
                 RETURN

        C        NORMAL CALL
              10 C=6.283185*SX*SY
                 XMOD=MOD
```

```
      B=AMOD/(A*A)
      N=MOD-1
      XJ=1.0
      TSUM=0.
      LL=0
      LLX = LL + 1
      NX = N+1
      DO 7 JX= LLX,NX
      J = JX - 1
      SUM=0.
      NN=J*(J+1)/2+1
      K=0
      KX = K + 1
      JY = J + 1
      DO 3 LX = KX,JY
      L = LX - 1
      NNN=NN+L
      Y1=G(2*L,SX,XMU,B,BIN,W)
      Y2=G(2*(J-L),SY,YMU,B,BIN,W)
      TERM=BIN(NNN)*Y1*Y2
      SUM=SUM+TERM
    3 CONTINUE
      TTERM=B**J*SUM
      IF(J)5,6,5
    5 TTERM=TTERM/XJ
      XJJ=J
      XJ=XJ*(XJJ+1.0)
    6 TSUM=TSUM+TTERM
    7 CONTINUE
      SSKP =TSUM/C
      RETURN
      END
```

FUNCTION G (DECK #9)

A. GENERAL

This function is used in conjunction with the SSKP function for elliptical errors with a normal offset weapon aim point. Its use is described with the SSKP function.

B. REQUIREMENTS IN CALLING PROGRAM

The requirements on the calling program are supplied by the function SSKP. They are the input parameters as described plus values for two arrays, BIN and W, used in computing binomial coefficients and the Gamma function.

C. ALGORITHMS IMPLEMENTED

As described in the SSKR description, the function is given by

$$H(M,\sigma,\mu,b) = \left(\frac{\beta}{\alpha}\right)^M \left(\frac{\sigma}{\alpha}\right) \exp\left[\frac{\beta^2-\mu^2}{2\sigma^2}\right] \cdot \sum_{j=0}^{M} \left(\frac{M}{2j}\right)\left(\frac{\sigma}{\beta}\right) 2^{(j+1)/2}\, \Gamma\left(\frac{j+1}{2}\right) .$$

The coefficient of the exponent term is different here than in the SSKP description. This follows the coding, while the other follows the write-up in Lambda Paper 6. The subroutine was not thoroughly checked for other possible differences.

```
      FUNCTION G(M,SIG,XM,HB,BIN,W)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 7, 1972
C     ONLY USED FOR SPECIAL CALCULATIONS FOR THE FUNCTION SSKP.
C     SEE LAMDA PAPER 6.

      DIMENSION BIN(1),W(11)
      ALPHA=SQRT (1.+2.*HB*SIG*SIG)
      SA=SIG/ALPHA
      IF(XM)11,10,11
   11 BETA=XM/ALPHA
      SB=SIG/BETA
      BA=BETA/ALPHA
      L=0
      SUM=0.0
      NN=M*(M+1)/2+1
      LX = L + 1
      MX = M + 1
      DO 1 KX= LX,MX,2
      K = KX - 1
      NNN=NN+K
      L2=K/2+1
      Z = K
      TERM=BIN(NNN)*SB**Z*W(L2)
    1 SUM=SUM+TERM
      P=EXP ((BETA*BETA-XM*XM)/(2.*SIG*SIG))
      V = M
      G=BA**V*SA*P*SUM
      RETURN
   10 G1=SA**(M+1)
      L3=M/2
      G=G1**(L3+1)
      RETURN
      END
```

FUNCTION XLRAD (DECK #10)

A. GENERAL

This function is used to obtain the distance at which certain blast overpressures occur. As such it is similar to PDIST. It is based upon numerical lists of pressure distance curves with the constants contained in several arrays. This function has been used at IDA for a number of years in evaluation programs.

Two options are available. A positive pressure as an argument yields a surface burst distance, a negative value yields the distance obtained when the height of burst is chosen to maximize the distances from ground zero at which the input pressure occurs.

B. REQUIREMENTS IN CALLING PROGRAM

The only requirement for this function is the input pressure. ITS MAGNITUDE ~~The mani~~ is used to find the distance. If the sign is positive a surface burst is implied, if negative an optimized air burst.

C. ALGORITHM IMPLEMENTED

The logarithm of the distance is found by LOGARITHMIC ~~linear~~ interpolation from the tabulated values.

PRECEDING PAGE BLANK-NOT FILMED

```
      FUNCTION XLRAD(Q)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 7, 1972
C     RETURNS PRESSURE GIVEN DISTANCE. BASED ON OLD WSEG FITS.
      DIMENSION T(13),X(13),Y(26)
      DATA  T( 1) /2000./
      DATA  T( 2) /1000./
      DATA  T( 3) /500./
      DATA  T( 4) /300./
      DATA  T( 5) /100./
      DATA  T( 6) /50./
      DATA  T( 7) /30./
      DATA  T( 8) /20./
      DATA  T( 9) /15./
      DATA  T(10) /10./
      DATA  T(11) /6./
      DATA  T(12) /3./
      DATA  T(13) /1./
      DATA X ( 1) /7.6/
      DATA X ( 2) /6.90776/
      DATA X ( 3) /6.21461/
      DATA X ( 4) /5.70378/
      DATA X ( 5) /4.60517/
      DATA X ( 6) /3.91202/
      DATA X ( 7) /3.40120/
      DATA X ( 8) /2.99573/
      DATA X ( 9) /2.70805/
      DATA X (10) /2.30259/
      DATA X (11) /1.79176/
      DATA X (12) /1.09861/
      DATA X (13) /0./
      DATA Y ( 1) /-2.659/
      DATA Y ( 2) /-1.561/
      DATA Y ( 3) /-1.273/
      DATA Y ( 4) /-1.05/
      DATA Y ( 5) /-.616/
      DATA Y ( 6) /-.301/
      DATA Y ( 7) /-0.062/
      DATA Y ( 8) /0.1222/
      DATA Y ( 9) /0.44469/
      DATA Y (10) /.77011/
      DATA Y (11) /1.15688/
      DATA Y (12) /1.71919/
      DATA Y (13) /2.2213/
      DATA Y (14) /-2.659/
      DATA Y (15) /-1.561/
      DATA Y (16) /-1.273/
      DATA Y (17) /-1.05/
      DATA Y (18) /-.616/
      DATA Y (19) /-.301/
```

```
      DATA Y (20) /-.062/
      DATA Y (21) /.1222/
      DATA Y (22) /.3001/
      DATA Y (23) /.49470/
      DATA Y (24) /.81978/
      DATA Y (25) /1.25276/
      DATA Y (26) /1.88403/
      B2=0
      XX=0
C     Q GT ZERO IS SURFACE BURST, LESS THAN ZERO OPT. AIR BURST.
      IF (XX.GE.0.)GO TO 1
      B2=13
      XX=-Q
1     TEMP1= XX
      IF (1. .GE.XX) TEMP1=1.001
      DO 2 II=1,13
      I=13-II+1
      IF (T(I) .GT. TEMP1) GO TO 3
2     CONTINUE
      I=12
3     TEMP2= X(I+1)-X(I)
      XXI = I
      B2 = B2 + XXI
      IB = B2
      IBP     = B2 + 1.
      S=(Y(IBP )-Y(IB))/TEMP2
      B=-(S*X(I+1))+Y(IBP )
      Z=ALOG(TEMP1)*S+B
      XLRAD = EXP(Z)
      RETURN
      END
```

FUNCTION ROOTF (DECK #11)

A. GENERAL

This function finds the root of ~~a~~ THE function specified in ~~a~~ THE calling sequence.

The search interval is specified, and the error in the function argument allowed.

If no root is found the function is exited with a function value are greater than the maximum allowable. If the function has more than one root usually the first root is found but this is no guarantee that this root is the one obtained.

B. REQUIREMENT IN CALLING PROGRAM

The calling program must supply for arguments for the function. They are:

XO	-	minimum value of interval searched
XF	-	maximum value of interval searched
EFF	-	error allowable in argument of function
DUMMY	-	the name of the function which is to have the root found

The function value provided is the value of the independent variable in the function which makes the value of the function equal to zero.

C. ALGORITHM IMPLEMENTED

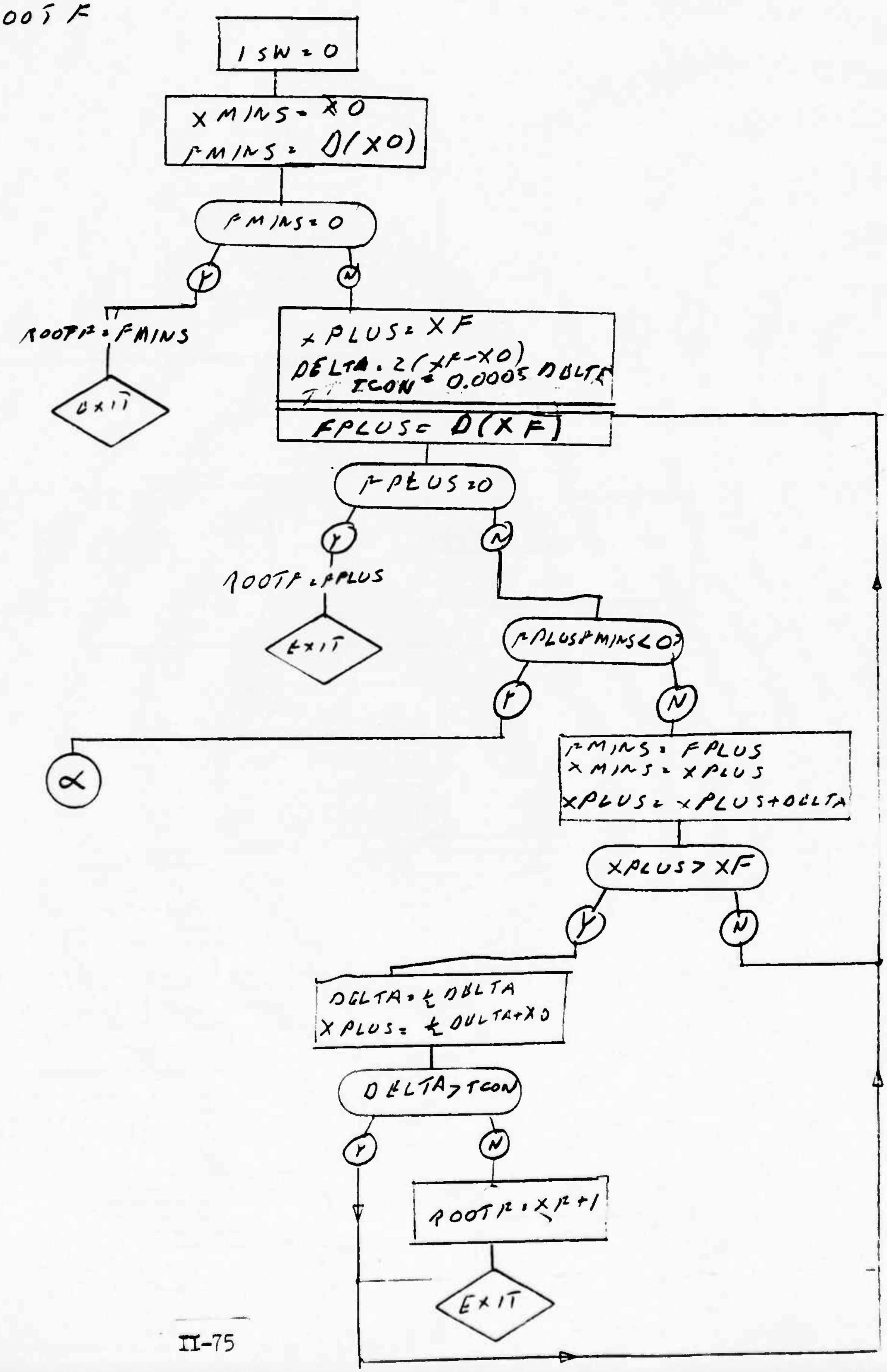
A flow chart is provided to assist following the logic.

The subroutine first attempts to find one positive and one negative function value. The two ends are first searched. If both ends have the same sign, the left end of the interval search is set equal to x0 and the right end to ~~left end~~ xo plus 1/2 the

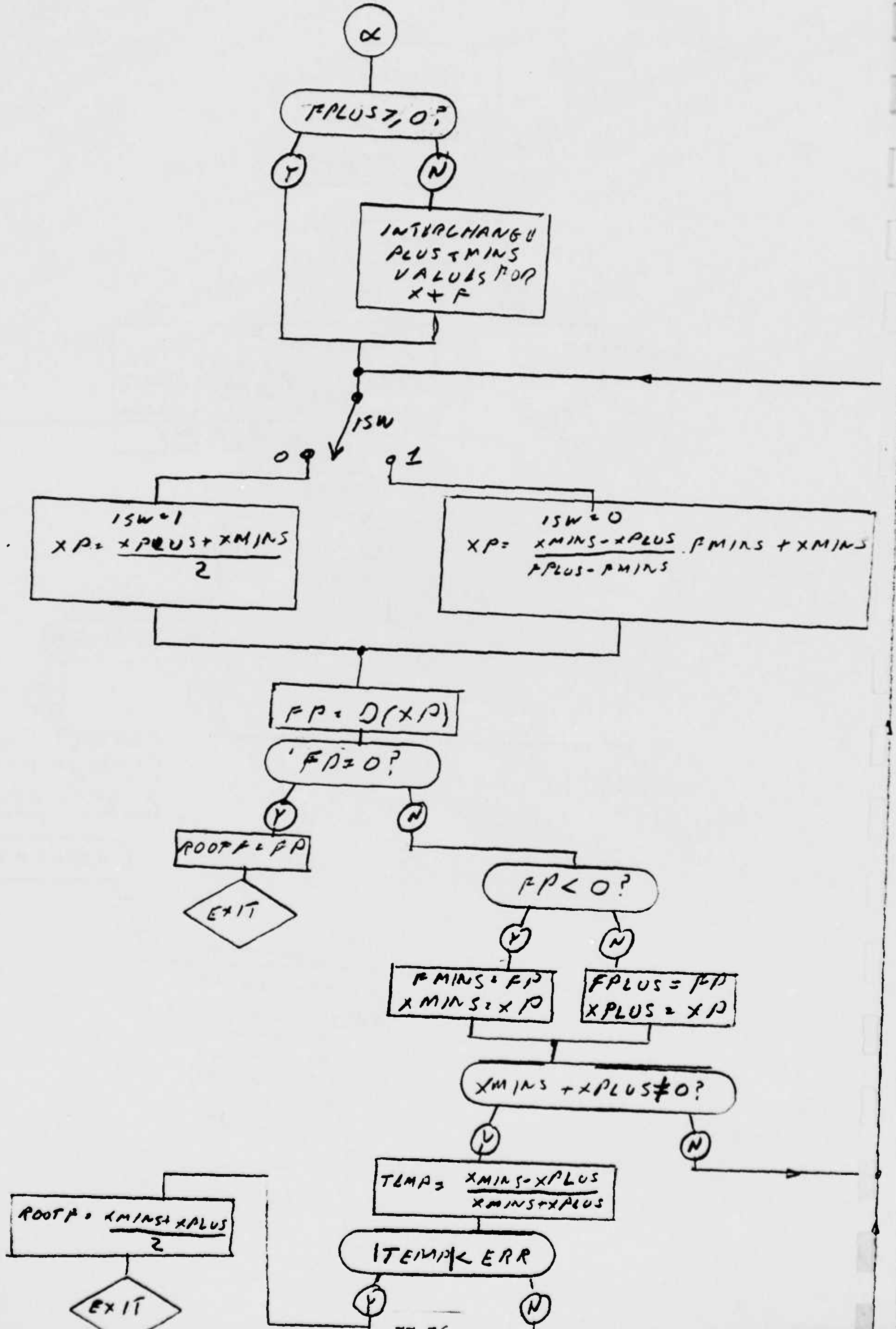
interval length, XF-X0 If the signs are the same the search interval is moved to the right and the process continued until the ends of the search interval yield different function value signs or the right end of the ALLOWED interval is reached. If the right end is reached the size of the search interval is ~~decreased~~ HALVED and the stepping process repeated until a good search interval is found. The search interval size is decreased to 1/1000th the initial interval size and if no alternate signs are found an error exit is made with the values of $XF*10^{15}$.

Once an interval with alternating sign is found one of two processes is used alternately to find a better estimate of the root. One method is linear interpolation between the end points of the interval. The other is to take for the new point the midpoint of the interval. A new interval is constructed with the new point and that end point which keeps the sign of the function at the two endpoints of the interval differential. The process is continued until the two endpoints differ by less than the value error. At this point the midpoint is chosen at the argument. If after 1000 trials the interval is still larger than EFF an error exit is taken with function values equal $XF*10^{16}$.

ROOT F


ISW = 0
XMINS = XO
FMINS = D(XO)
FMINS = 0
Y
N
ROOTF = FMINS
EXIT
XPLUS = XF
DELTA = 2(XF-XO)
TCON = 0.0005 DELTA
FPLUS = D(XF)
FPLUS = 0
Y
N
ROOTF = FPLUS
EXIT
FPLUS*FMINS<0?
Y
N
α
FMINS = FPLUS
XMINS = XPLUS
XPLUS = XPLUS+DELTA
XPLUS > XF
Y
N
DELTA = ½ DELTA
XPLUS = ½ DELTA+XO
DELTA > TCON
Y
N
ROOTF = XF+1
EXIT

ROOTF p.2



α
FPLUS ≥ 0?
Y
N
INTERCHANGE PLUS + MINS VALUES FOR X + F
ISW
0
1
ISW = 1
XP = (XPLUS + XMINS)/2
ISW = 0
XP = (XMINS - XPLUS)/(FPLUS - FMINS) · FMINS + XMINS
FP = D(XP)
FP = 0?
Y
N
ROOTF = FP
EXIT
FP < 0?
Y
N
FMINS = FP
XMINS = XP
FPLUS = FP
XPLUS = XP
XMINS + XPLUS ≠ 0?
Y
N
TEMP = (XMINS - XPLUS)/(XMINS + XPLUS)
\|TEMP\| < ERR
Y
N
ROOTF = (XMINS + XPLUS)/2
EXIT

```
          FUNCTION ROOTF(X0,XF,ERR,DUMMY)

      C   NEVUNS STANDARD
      C   LAST REVISED NOV. 7, 1972

      C   A FUNCTION WHICH RETURNS AS A FUNCTION VALUE THE FIRST ROOT ,
      C   USUALLY OF THE FUNCTION DUMMY.
      C   USUALLY, OF THE FUNCTION DUMMY.

          KNT = 0
          ISW=0
      C   INITITALIZE XMINUS
          XMINS=X0
          FMINS=DUMMY(XMINS)
          ROOTF=XMINS
          IF(FMINS.EQ.0.)RETURN
          XPLUS=XF
          DELTA = (XF-X0)*2.
          TCON=DELTA*.0005

      C   STEP THROUGH UNTIL STRADDLE ROOT.
       1  FPLUS=DUMMY(XPLUS)
          ROOTF=XPLUS
          IF(FPLUS.EQ.0.)RETURN
          IF(FPLUS*FMINS.LT.0.)GO TO 2
          FMINS=FPLUS
          XMINS=XPLUS
          XPLUS = XPLUS + DELTA
          IF(XPLUS-XF.LT.0.)GO TO 1
      C   DECREASE DELTA AND START OVER LOOKING FOR ROOT.
          DELTA=DELTA*.5
          XPLUS=DELTA*.5+X0
          IF(DELTA-TCON.GE.0.) GO TO 1
      C   ERROR EXIT
       3  ROOTF = XF + 1.E+15
          RETURN
       2  IF(FPLUS .GE. 0.) GO TO 11
          TEMP=XPLUS
          XPLUS=XMINS
          XMINS=TEMP
          TEMP=FPLUS
          FPLUS=FMINS
          FMINS=TEMP
       11 IF(ISW .EQ. 0) GO TO 12
          ISW=0
      C   ILINR-LINEAR INTERPOLATION
          T1=FPLUS-FMINS
          XP=(XMINS-XPLUS)*FMINS/T1+XMINS
          GO TO 13
      C   IBNRY-BINARY DIVIDE
       12 XP=(XPLUS+XMINS)*.5
          ISW=1
       13 FP=DUMMY(XP)
          ROOTF=XP
          IF(FP .EQ. 0.)RETURN
```

```
      IF( FP .LT. 0.) GO TO 14
      FPLUS=FP
      XPLUS = XP
      GO TO 15
   14 FMINS=FP
      XMINS=XP
   15 IF(XMINS-XPLUS .EQ. 0.) GO TO 11
      KNT = KNT + 1
      IF( KNT .LT. 1000) GO TO 17
C     ERROR EXIT NO CONVERGENCE
      ROOTF = XP + 1.E+16
      RETURN
   17 CONTINUE
      TEMP = XMINS - XPLUS
      IF(TEMP .GE. 0.) GO TO 16
      TEMP = - TEMP
   16 IF (TEMP-ERR .GE.0.)GO TO 11
      ROOTF=(XMINS+XPLUS)*.5
      RETURN
      END
```

SUBROUTINE CALRN

A. GENERAL

The purpose of this subroutine is to collect together ~~the~~ various random number generating functions. Since the method of obtaining random numbers is not uniform between systems, the subroutine simplifies ~~the method of~~ transferring between machines.

The subroutine provides for initialization of the random number generation process and returns a number sampled from a uniform, exponential or normal distribution.

B. REQUIREMENTS ON CALLING PROGRAM

Communication between the subroutine and the calling program is through the common block TRAN1 which has values RNO as input and RNW as the generated random number.

Values of RNO control the process as follows:

RNO	greater than zero but not equal 1. The value of RNO is used as the seed for generating a string of numbers
RNO=1	The seed is generated by reading the computer clock. This calling routine is 6400 unique. For machines without a clock other means of generating a "random" seed are needed. Such procedures are, of course, unique within a system. One of the previous two calls must be made to initialize the random number generator. The calls leave RNW undefined.
RNO=0	A random floating print number sampled from a uniform distribution in (0,1) is picked.

RNO=-1 A number sampled from an exponential distribution will mean = 1 is returned.

RNO less than zero but not equal -1

A number sampled from a normal distribution will mean 0 and standard deviation = 1 is returned.

The subroutine uses the following subroutines or functions defined only for the CDC 6400 FTN computer. They must be replaced by the appropriate analogs for other systems.

Subroutine RANSET(X)	-	the seed of the random number generator is set equal to X
Fundion RANF(0)	-	a floating point random number uniformly distributed in (0,1) is returned
Subroutine TIME(CLTIM)	-	the software clock is read and returns the value CLTIM.

C. ALGORITHMS IMPLEMENTED

The uniform distribution is obtained directly from the CDC system function which uses a con~~sequ~~GRUential method to generate the random number.

For the exponential distribution a random number μ is drawn from a uniform distribution. The number returned, RNW, is given by

$$RNW = -\ln \mu$$

If the initial number is drawn from a uniform distribution the method is exact.

~~To~~ FOR the normal distribution the algorithm is taken from a recent ACM article by Ahrens and Dieter[1] which is reproduced here, in part.

1. Ahrens, J. H., and U. Dieter "Computer Methods for Sampling from the Exponential and Normal Distributions", Communications of the Association for Computing Machinery, Oct. 1972, Vol. 15, No. 10, pp. 873-882.

The method approximates the normal curve by a trapazoid for most calls, but for some calls adds corrections. The method is exact if the random number generator samples form a true uniform distribution, AND IS MORE RAPID THAN THE SIMPLE SUMMING OF SIX OF UNIFORMLY DISTRIBUTED RANDOM NUMBERS.

The FORTRAN program for CT was rather slow, it required [illegible] μsec per sample. The method came into its own in machine code only where the absence of functions (except for the square root in 15.73 percent of all cases) made for efficient coding. The assembler program (including the in-line square root) was 86 words long and required 374 μsec per sample. It was therefore as fast as but shorter than the modified polar method.

The next algorithm uses sampling from a convenient approximation to the normal probability density function with high probability. Marsaglia and Bray [16] describe a similar procedure in which three uniform samples are required in 86.38 percent of all cases, two such samples are taken with a probability of 11.07 percent, and in the remaining 2.55 percent of all cases, an acceptance-rejection technique or a tail method is used.

In the subsequent "trapezoidal" algorithm (TR) two uniformly distributed variables are needed in 91.95 percent of all cases. The procedure is easy to program in high-level languages but is not suitable for machine coding. It has moderate performance characteristics similar to the method of Marsaglia and Bray [16].

The trapezoidal method TR requires, as a preparation, the solution of a numerical exercise: a maximum trapezoid is to be inscribed in the graph of the standard normal probability density function. This largest possible trapezoid T is given by its vertices

$(\pm\xi, 0)$ where $\xi = 2.11402\ 80833\ 3742$

and

$(\pm X, Y)$ where $X = 0.28972\ 95736$

and $\quad Y = 0.38254\ 45560\ 42518.$

T covers $A = 0.91954\ 44057\ 06926$ of the total area (which is 1) under the standard normal curve.

Sampling from a (symmetrical) trapezoid is easy: if u_1 and u_2 are two independent uniform variables, then any combination $c_1u_1 + c_2u_2$ ($c_1, c_2 > 0$) follows a distribution whose probability density function is shaped like an isosceles trapezoid. In step 1 of the detailed description the constants are worked out (u_0 is uniform in (0, 1) and u is uniform in $(0, A)$). This step applies in $A \approx 91.95$ percent of all cases.

With a probability of about 3.45 percent the tail method TL is used in step 3 for all samples beyond $\pm\xi$.

The steps 5, 7 and 8 are acceptance-rejection procedures (AR) on the difference function between $f(x)$ and the trapezoid in different intervals between $\pm\xi$. The steps 3, 5, 7 and 8 deliver samples y from the signless distribution $\varphi(y)$ so that in step 9 a random sign has to be attached.

Algorithm TR (Trapezoidal method, Ahrens)

1. Generate u and u_0. If $u < 0.91954\ 44057\ 06926$, deliver $x \leftarrow 2.40375\ 76569\ 3742 \times (u_0 + u \times 0.82535\text{-}92825\ 36923) - 2.11402\ 80833\ 3742$.
2. If $u < 0.96548\ 71312\ 13858$ go to 4.
3. Generate u_1 and set $y \leftarrow (4.46911473713927 - 2 \ln u_1)^{\frac{1}{2}}$. Generate u_2, and if $yu_2 > 2.11402808333724$ then repeat this step 3. Otherwise go to 9.
4. If $u < 0.94999\ 07087\ 33028$ go to 6.
5. Generate u_1 and set $y \leftarrow 1.84039874739771 + u_1 \times 0.27362\ 93359\ 39706$. Generate u_2, and if $0.398942280\text{-}401433 \exp(-y^2/2) - 0.44329\ 91258\ 20220 + y \times 0.20969\ 40571\ 95486 < u_2 \times 0.042702581590795$ then repeat this step 5. Otherwise go to 9.
6. If $u < 0.925852333707704$, go to 8.
7. Generate u_1 and set $y \leftarrow 0.28972\ 95736\ 00000 + u_1 \times 1.55066917379771$. Generate u_2, and if $0.398942280\text{-}401433 \exp(-y^2/2) - 0.443299125820220 + y \times 0.20969\ 40571\ 95486 < u_2 \times 0.01597\ 45226\ 55238$ then repeat this step 7. Otherwise go to 9.
8. Generate u_1 and set $y \leftarrow u_1 \times 0.289729573600000$. Generate u_2, and if $0.398942280401433 \exp(-y^2/2) - 0.382544556042518 < u_2 \times 0.016397724358915$ then repeat this step 8. Otherwise continue with 9.
9. Use u_0 to determine the sign of the sample: if $u_0 < 0.5$, then deliver $x \leftarrow y$. Otherwise deliver $x \leftarrow -y$.

The method was programmed in FORTRAN only, and there it proved faster than any other algorithm described in this section: an average sample required 383 μsec.

It was estimated that TR would require about 300 μsec if it were programmed in assembler code. The similar method in Marsaglia and Bray [16] should be as fast as TR in FORTRAN and slightly better than TR (270 μsec) in assembler code. Both algorithms use transcendental functions which impose a space penalty if they are not required in other parts of the program.

Polynomial sampling, the essence of the next method, uses no standard functions and yields extra flexibility since one may choose faster programs with larger tables of coefficients or shorter, but slower, procedures. The polynomial method (TS), given below, for the standard normal distribution is faster (in assembler code) than any previously listed method in this paper. It requires about as much space as the IBM 360 routine for the gamma function which uses as much memory as the logarithm and the exponential functions combined.

The area under the graph of $\varphi(y)$ (cf. (4.3)) is split into five parts: 1st, 2nd, 3rd, 4th, and Tail with areas 1/2, 1/4, 1/8, 1/16, and 1/16 respectively. For each of the first four parts a table has to be provided following the prescriptions in the procedure PL of Section 2. The Taylor expansions at the four left interval boundaries x_0 are used; their coefficients can be worked out easily by computer using the differential equation $\varphi' = -y\varphi$. Table T contains the resulting values of x_0, a, t, s_i, and P_i (in the notation of the procedure PL in Section 2).

[illegible] Normal Distribution.

[illegible]	N	IBM 360/50 FORTRAN	IBM 360/50 Assembler	IBM 360/50 Core	CDC 6400 Assembler	CDC 6400 Core
[illegible] (S[illegible]-Cosine)	1.00000	1093 μsec	—	—	—	—
[illegible] PO (Polar)	1.27324	903 μsec	—	—	—	—
[illegible] (Mod. Polar)	2.11981	791 μsec	384 μsec	127W	92 μsec	61W
[illegible] (Center-Tail)	6.52932	1154 μsec	374 μsec	86W	89 μsec	40W
[illegible] (Trapezoidal)	2.33066	583 μsec	—	—	—	—
6. TS (Taylor-Series)	1.67894	707 μsec	229 μsec	213W	56 μsec	160W
7. TS'	1.26874	680 μsec	210 μsec	274W	—	—
8. TS''	1.28387	663 μsec	202 μsec	295W	—	—
9. RT (Rect.-Wedge-Tail)	1.28168	647 μsec	168 μsec	436W	—	—

(MT) based on Table W, choosing the appropriate accuracy of the computer in hand. For the uniform variable u in step 1 of MT use the existing u from above. Since U < 8 this u is smaller than 8/256 = 1/32. (This [illegible] Table W in which the sum of the probabilities is [illegible].) If the result is the tail (13), then go to 6. Otherwise [illegible] W is obtained.

[illegible] polynomial sampling (PL') based on Table [illegible] for the [illegible] wedge W_j. This yields a sample y. Go to 7.

[illegible] Marsaglia's tail method modified by the Algorithm SA as described under TA (Table 2 under Algorithm TS is needed.) The result is a sample y from the [illegible] ($\xi = 3.0$, $\xi^2/2 = 4.5$.)

[illegible] the sample y from 2, 5, or 6 the sign [illegible] stored in step 1. The result is delivered as the [illegible] sample x from $f(x)$.

[illegible] programs needed 647 μsec in FORTRAN [illegible] assembler. The core requirements were [illegible] M, 78 words for the tables resulting [illegible] Marsaglia algorithm (step 4), 141 words for the constants required in the polynomial sampling procedure (step 5) and 10 words for the Table 2 in step 6. [illegible] these tables consumed 291 words out of a total [illegible] 436 words for the complete assembler [illegible]. The algorithm is marginally faster than the [illegible] version of Marsaglia et al. It also needs more [illegible] for tables but uses no standard-function subprograms.

The [illegible] computational results for the entire series of programs for the normal distribution are now summarized (see Standard Normal Distribution.) PA, CT, and TS were also tested in CDC-6400 Compass code.

The authors recommend SC or PO only if convenience of programming is the main consideration. [illegible] a method like TR is good for high level [illegible] whereas machine code utility routines should be based on CT, TS, or RT depending on the amount of [illegible] can be afforded.

Added in proof. Algorithm CT was speeded up by 10 percent (with N = 4.87889). It also gave rise to several [illegible] algorithms by G.E. Forsythe and the authors that [illegible] well with TS and RT. They will be published [illegible] *[illegible] of Computation* and in a book (Springer) on random numbers by the authors.

Acknowledgment. The editorial help by Roman L. Weil is gratefully acknowledged.

Received June 1971; revised August 1971

References

1. Ahrens, J.H., Dieter, U., and Grube, A. Pseudo-random numbers: A new proposal for the choice of multiplicators. *Computing 6* (1970), 121–138.
2. Box, G.E.P., and Muller, M.E. A note on the generation of random normal deviates. *Ann. Math. Stat. 29* (1958), 610–611.
3. Dieter, U. Autokorrelation multiplikativ-erzeugter Pseudo-Zufallszahlen. *Operations Res. Verfahren 6* (1969), 69–85.
4. Dieter, U. Pseudo-random numbers: The exact distribution of pairs. *Math. Comput. 25* (1971), 855–883.
5. Dieter, U. Pseudo-random numbers: Permutations of triplets. J. Res. Nat. Bureau Standards 76B (Jan.-June 1972).
6. Dieter, U., and Ahrens, J. An exact determination of serial correlation of pseudo-random numbers. *Numer. Math. 17* (1971), 101–123.
7. Dieter, U., and Ahrens, J. A combinatorial method for the generation of normally distributed random numbers. To appear.
8. Jöhnk, M.D. Erzeugen und Testen von Zufallszahlen. Berichte aus dem Institut für Statistik und aus dem Institut für Angewandte Statistik der FU Berlin, Heft 6, Physica-Verlag Würzburg, 1969.
9. Knuth, D.E. *The art of computer programming, Vol. II: Seminumerical Algorithms.* Addison Wesley, Reading, Mass., 1969.
10. MacLaren, M.D., Marsaglia, G., and Bray, T.A. A fast procedure for generating normal random variables. *Comm. ACM 7*, 1 (Jan. 1964), 4–10.
11. MacLaren, M.D., Marsaglia, G., and Bray, T.A. A fast procedure for generating exponential random variables. *Comm. ACM 7*, 5 (May 1964), 298–300.
12. Marsaglia, G. Expressing a random variable in terms of uniform random variables. *Ann. Math. Stat. 32* (1961), 894–899.
13. Marsaglia, G. Generating exponential random variables. *Ann. Math. Stat. 32* (1961), 899–902.
14. Marsaglia, G. Random variables and computers. Trans. Third Prague Conf., 1962, pp. 499–512.
15. Marsaglia, G. Generating a variable from the tail of the normal distribution. *Technometrics 6* (1964), 101–102.
16. Marsaglia, G., and Bray, T.A. A convenient method for generating normal variables. *SIAM Rev. 6* (1964), 260–264.
17. von Neumann, J. Various techniques in connection with random digits. Monte Carlo methods. *Nat. Bureau Standards, AMS 12* (1951), 36–38.
18. Sibuya, M. Exponential and other random variable generators. *Ann. Inst. Stat. Math. 13* (1961/62), 231–237.

```
          SUBROUTINE CALRN
C         NEVUNS STANDARD
C         LAST REVISED NOV. 20, 1972

C... THE PURPOSE OF THIS SUBROUTINE IS TO PUT RANDOM NUMBER
C         GENERATOR CALLS IN ONE PLACE
C         THIS WILL ALLOW CONVERTING TO OTHER SYSTEMS WITH A MINIMUM OF PAIN
C... IF RNO = ZERO GET A NEW RANDOM NUMBER. -INSERT INTO RNW
C         IF RNO IS POSITIVE OBTAIN SEED FOR RANDOM NUMBERS. THIS MUST
C         BE DONE BEFORE THE FIRST RANDOM NUMBER IS GENERATED.
C         IF RNO = 1. OBTAIN SEED (RANDOMLY) BY READING CLOCK,OTHERWISE
C         USE RNO AS VALUE OF SEED.
C         IF RNO IS -1 DRAW NUMBER FROM EXPONENTIAL DISTRIBUTION, ANY OTHER
C         NEGATIVE NUMBER DRAW FROM NORMAL DISTRIBUTION, N(0,1)

          COMMON /TRAN/RNO,RNW

          IF(RNO .NE. 0.) GO TO 10

C         GET A RANDOM NUMBER UNIFORMLY DISTRIBUTED FROM 0 TO 1.FLOATING PT.
C         RANF(0) IS UNIQUE TO CDC 6400 FTN
          RNW = RANF(0)
          RETURN

10        CONTINUE
          IF (RNO .LT. 0.)GO TO 20
          IF(RNO .EQ.1.)GO TO 15

C         INITIALIZE RANDOM NUMBER GENERATOR WITH RNO AS SEED
C         RANSET IS UNIQUE TO CDC 6400 FTN
          CALL RANSET(RNO)
          RETURN

C         READ THE SOFTWARE CLOCK TO OBTAIN SEED HOPEFULLY AT RANDOM.
C         THE FUNCTION TIME IS CDC 6400 FTN UNIQUE.
15        CONTINUE
          CALL TIME(CLTIM)
          CLTIM = ABS(CLTIM)
          RNI = CLTIM + 1.2
          CALL RANSET(RNI)
          RETURN

20        CONTINUE
          IF(RNO .NE. -1.) GO TO 40
C         DRAW NUMBER FROM EXPONENTIAL DISTRIBUTION WITH MEAN = 1
          U = RANF(0)
          RNW = - ALOG(U)
          RETURN
40        CONTINUE
```

```
      C     DRAW NUMBER FROM NORMAL DISTRIBUTION WITH MEAN 0 AND STD. DEV. 1
      C     USE AS METHOD ALGORITHM TR FROM #COMPUTER METHODS FOR SAMPLING
      C     FROM THE EXPONENTIAL AND NORMAL DISTRIBUTIONS#, J. H. AHERNS
      C     AND U. DIETER, COMMUNICATIONS OF THE ACM, OCT., 1972, VOL. 15,
      C     NO. 10, P. 873.

      41    CONTINUE
            U = RANF(0)
            U0 = RANF(0)
            IF(U .GE. 0.919544405706926 ) GO TO 42
            RNW = 2.403757656937 42*(U0 + U*0.825339282536923) -
           1 2.11402808333742

      42    CONTINUE
            IF( U .LT. 0.965487131213858) GO TO 44

      43    CONTINUE
            U1 = RANF(0)
            Y = 4.46911473713927 - 2.*ALOG(U1)
            U2 = RANF(0)
            IF( Y*U2 .GT. 2.11402808333724) GO TO 43
            GO TO 49

      44    CONTINUE
            IF(U.LT. 0.949990708733028) GO TO 46

      45    CONTINUE
            U1 = RANF(0)
            Y = 1.84039874739771 + U1*0.273629335939706
            U2 = RANF(0)
            TEMP = 0.398942280401433*EXP(-Y*Y/2.) - 0.443299125820220
           1 + Y*0.209694057195486
            IF( TEMP .LT. U2*0.042702581590795) GO TO 45
            GO TO 49

      46    CONTINUE
            IF( U .LT. 0.925852333707704 ) GO TO 48

      47    CONTINUE
            U1 = RANF(0)
            Y = 0.289729573600000 + U1*1.55066917379771
            U2 = RANF(0)
            TEMP =  0.398942280401433*EXP(-Y*Y/2.) - 0.443299125820220
           ) + Y* 0.209694057195486
            IF( TEMP .LT. U2*0.015974522655238) GO TO 47
            GO TO 49

      48    CONTINUE
            U1 = RANF(0)
            Y = U1* 0.289729573600000
            U2 = RANF(0)
            TEMP = 0.398942280401433*EXP(-Y*Y/2.) - 0.382544556042518
            IF(TEMP .LT. U2* 0.016397724358915) GO TO 48
            GO TO 49
```

```
   49     CONTINUE
          IF( U0 .LT. 0.5) GO TO 50
          RNW = -Y
          RETURN
   50     CONTINUE
          RNW = Y
          RETURN
          END
```

FUNCTION CUMNOR (Deck #12)

A. GENERAL

This function computes the cumulative normal function

$$y(x)^2 \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \quad .$$

It is based on a formula in Hastings[1] on page 169 which has a maximum error of 0.00000015. A faster formula may be found on page 187 which does not require computing the exponential function.

B. REQUIREMENTS ON THE CALLING PROGRAM

The calling program furnishes a value of argument x which is normally distributed with mean 0 and variance one. The function value is the cumulative distribution function. If $x < 7$ a value of 0.0 is returned. If $x > 7$ a value of 1.0 is returned.

C. ALGORITHMS IMPLEMENTED

Let

$$T = \left(\frac{1}{1+0.2316419 \cdot |x|}\right)$$

$$D = .03989432 \exp(-0.5x^2) \ .$$

Then

$$P=1.0-D \cdot T((((1.330274T-1.821256)T + 1.781478)T$$
$$-0.3565638)T + 0.3193815) \ .$$

---

1. Hastings, Cecil, "Approximations for Digital Computers," Princeton University Press, 1955.

If $x < 0$ then set $P = 1-P$

If $x < 7$ $\qquad$ $P = 0.0$

If $x > 7$ $\qquad$ $P = 1.0$

```
      FUNCTION CUMNOR (X)
C     NEVUNS STANDARD
C     LAST REVISED NOV. 7, 1972
C     COMPUTES Y = P(X) = PROBABILITY THAT THE RANDOM VARIABLE U,
C     DISTRIBUTED NORMALLY(0,1), IS LESS THAN OR EQUAL TO X.
C     IF X IS GREATER THAN 7  P = 1
C     IF X IS LESS THAN -7  P = 0
C     SEE HASTINGS APPROX FOR DIGITAL COMPUTEERS, P. 169
C     BETTER IS HASTINGS P. 187 WHICH DOES NOT HAVE TO USE THE EXP
C     FUNCTION. SEE FUNCTION CUMNOA
      AX = ABS(X)
      IF (AX - 7.0) 10,10,20
   10 CONTINUE
      T = 1.0/(1.0+.2316419*AX)
      D = 0.3989423*EXP(-0.5*X*X)
      P = 1.0 - D*T*((((1.330274*T - 1.821256)*T + 1.781478)*T -
     A    0.3565638)*T + 0.3193815)
      IF (X) 1,2,2
    1 P = 1.0 - P
    2 CUMNOR = P
      RETURN
   20 IF (X) 30,30,40
   30 CUMNOR=0.
      RETURN
   40 CUMNOR=1.0
      RETURN
      END
```

SUBROUTINE FORD(Deck #210)

A. GENERAL

This subprogram orders elements in an array in increasing numerical order. The subprogram uses a very simple ordering technique, and is much slower than other methods. It was written to be a simple subprogram written strictly in FORTRAN for ordering arrays which are not too lengthy.

B. REQUIREMENTS ON CALLING PROGRAM

The external communications are augmented in the calling of the subroutine. They are

ARR(I)--the array to be ordered. This array is treated as a floating point number between $-\infty$ and ∞ and numbers are ordered algebraically.

LARR(i)--index to the ordered numbers. The numbers in ARR are not physically rearranged but the array LARR gives the place in ARR for increasing values in ARR, with the first entry in LARR referring to the lowest number. Thus ARR(LARR(1)) is the smallest element, ARR(LARR(2)) the next smallest, and ARR(LARR(NITM)) the largest element.

NITM--the number of elements to be ordered. It is the calling program responsibility to insure NITM is a correct value since the subprogram has no error responses.

C. ALGORITHM IMPLEMENTED

The ordering is very simple. Suppose the first i things in LARR have been ordered. If the $i+1^{st}$ item is larger than ARR(LARR(i)) then LARR(i+1) = i + 1. If not, then a search is

made through the first i elements in increasing order (using LARR(i)) until a larger item is found. Then it and all higher values of ARR are pushed down one in LARR, and the $i+1^{st}$ item inserted in LARR.

```
      SUBROUTINE FORD(ARR,LARR,NITM)
      DIMENSION ARR(101),LARR(101)
      IT = 1
      LARR(IT) = 1
      VAL = ARR(IT)
10    CONTINUE
      IT = IT + 1
      IF(IT .GT. NITM) GO TO 100
      IF(ARR(IT) .LT. VAL) GO TO 12
      LARR(IT) = IT
      VAL = ARR(IT)
      GO TO 10
12    CONTINUE
      JK = IT - 1
      DO 13 J = 1,JK
      JJ = J
      ITM = LARR(J)
      IF(ARR(ITM).LT. ARR(IT)) GO TO 13
      GO TO 14
13    CONTINUE
14    CONTINUE
      IK = IT - JJ
      DO 15 K = 1,IK
      KK = IT - K
      KKK = KK + 1
      LARR(KKK) = LARR(KK)
15    CONTINUE
      LARR(JJ) = IT
      GO TO 10
1'    CONTINUE
      RETURN
      END
```

SUBROUTINE PROJCT (Deck # 13)

A. GENERAL

This subroutine transforms from latitudes and longitudes to rectangular coordinates. A control parameter determines whether the output is in a rectangular coordinate system in statute or nautical miles, a grid system, or in inches for map plotting. Another control parameter determines the type of projection used. A number of standard map projection schemes are implemented to enable plotting overlays for a variety of maps. In addition, a projection to a set of rectangular coordinates is given which reduces distortion in distance calculation due to earth sphericity to at most a few percent. This is done though at the expense of true North deviating from North in this coordinate system by up to 15°. An option allows the grid system to be used for this projection which in effect replaces the latitude-longitude system with one or more appropriate for approximate distance calculations in the United States.

To simplify use, the control of the subroutine is the same for each projection although different data items may have to be supplied for each projection. A value of control parameter JC=2 uses default values of most input constants, a value = 3 requires the user to supply most values. In normal use the subroutine is first called with JC = 2 or 3 to initialize it, and then called with JC = 1 to get coordinate values from the input latitude and longitude.

B. REQUIREMENTS ON CALLING PROGRAM

The communications with the calling program are through the common block /CARTOG/. These variables define what is desired of the calling program.

IC - Control for type of projection

IC = 1 Flat earth distance elements from sphere
= 2 Flat earth distance elements from ellipsoid
= 3 Rotate coordinate system to get projection error as if on the equator looking sideways at the earth
= 4 As 3 but also stretch coordinates to minimize distance errors
= 5 Albers equal area projection with two standard parallels on ellipsoidal earth
= 6 Albers equal area projection with two standard parallels on spherical earth
= 7 Mercator projection on ellipsoidal earth
= 8 Mercator projection on spherical earth
= 9 Lambert conformal conic projection with two standard parallels on ellipsoidal earth
= 10 Transverse mercator projection on ellipsoid

= 11 Universal transverse mercator projection
= 12 American polyconic projection on ellipsoidal earth
= 13 Trapazoidal projection on spherical earth
= 14 Sterographic horizon projection on spherical earth

JC - Control for type of action for this call. A call with JC = 2 or 3 (except when IC = 12) is always needed to initialize.

JC = 1 Find coordinates of one point
= 2 Initialize using prestored data
= 3 Initialize using data supplied in /CARTOG/; Requirements for supplying data are described more fully below.

KC - Control for type of output.

KC = 1 Output in rectangular coordinates in statute miles
= 2 Output in grid coordinates of 50 statute mile intervals
= 3 Output in inches for maps centered on FLATC, FLONC
= 4 Output in rectangular coordinates in nautical miles
= 5 Output in grid coordinates of 60 nautical mile intervals
= 6 Output in inches for maps centered on CRMPLA, and CRMPLO for latitude and longitude with coordinate system centered on FLATC, FLONC

FLATP Input latitude (degrees) for a point to be plotted Always needed if JC = 1

FLONP Input longitude (degrees) for a point to be plotted Always needed if JC = 1

Y Output ordinate in rectangular coordinate system, (the meaning depends on the value of KC)

X Output abascissa in rectangular coordinate system, (the meaning depends on the value of KC)

RHO Polar coordinate radius output in statute miles, defined for conic projections. If IC = 11, output of Y coordinate in meters

THETA Polar coordinate angle output in radians, defined for conic projections. If IC = 11, output of X coordinate in meters

FLATC Latitude of center of coordinate system (degrees). Always needed if JC = 3. If JC = 2, FLATC is needed if IC is 1, 2,7,8,10,12,13 or 14. When KC = 1 or 4, distances are measured from the point defined by FLATC, FLONC.

FLONC Longitude of center of coordinate system (degrees) Needed as with FLATC

SCALE	Map scale factor. Distances on earth are divided by scale to obtain map distances. Needed if JC ≠ 1 and KC = 3 or 6.
STRCHUR	Paper stretch factor. North-South distances are multiplied by STRCHUR to compensate for paper stretching. Needed if JC = 3 and KC = 3 or 6. In standard use this value is near 1.
STCHAR	As STRCHUR but in East-West direction
HILATR	Latitude of highest latitude standard parallel (degrees) Needed if JC = 3 and IC = 5, 6 or 9. If IC = 11 this parameter has a different use. Its value is used to select the spheroid. See UTM description.
ROLATR	Latitude of lowest latitude standard parallel (degrees) Needed if JC = 3 and IC = 5, 6 or 9. If IC = 11 this parameter has a different use. If ROLATR = 0 the zone selected is that one containing FLONC, otherwise it specifies the zone number. See UTM description.
XGDSTR	Number of statute miles to subtract from center of coordinate system for EW origin of grid. Needed if JC = 3 and KC = 2 or 5, i.e., location of Western corner of grid in statute miles in coordinate system with KC = 1.
YDGSTR	Same as XGDSTR but in North-South direction
XMPOFR	Inches right offset for pen in plotting maps. XMPOFR = -15 means the pen starts 15" to the left of the center of the map. Needed for large scale maps of small areas. Needed for JC = 3 and KC = 3 or 6.
YMPOFR	Same as XMPOFR but in upwards direction.
CRMPLA	Latitude (degrees) of center of map. Used when the center of the map is different from the center of the coordinate system. Especially used for large scale maps of small areas. Needed if KC = 6.
CRMPLO	Same as CRMPLA but in longitude.

Besides the standard FORTRAN functions the subroutine requires a function ASIN which returns a value of angle in radians which is the arcSIN of an input value between 0 and 1.

C. ALGORITHMS IMPLEMENTED

The algorithms to compute the values of X and Y from the center of the coordinate system are described below. Those items which are common for every input-point are precomputed in initialization calls. In addition, some values

for specific output types are precomputed in the initialization. To be consistent with most standard projections the calculations assume a Clark Spheroid of 1866, except for IC = 11. If other spheroids are desired the constants CLARK for Earth Radius in statute miles, ECC for eccentricity and ECCSQ for eccentricity squared, STMID for number of statute miles per degree of latitude, AIMES for CLARK (1-ECCSQ), ECCTRO for ECCSQ/(1-ECCSQ) and ECCH for ECC/2, defined in data statements, should be changed.

The value of the earth radius is given in statute miles. This is obtained by dividing the values in meters used in the standard definition by the number of meters in a statute mile. This latter value is obtained by using the legal definition of a meter as exactly 39.37 inches. The conversion to nautical miles is obtained using the value of 6080.27 feet/nautical mile. If the subroutine is exercised with IC = 2 and FLATP one minute larger than FLATC, the result is the number of statute miles in a nautical mile if a nautical mile has the alternative definition as the length of a minute of arc at the latitude of interest. Values for other spheroids are given in the UTM discussion.

<u>FLAT EARTH</u> (IC = 1 or 2)

For an assumed flat earth the x and y coordinates are computed by

$$x = C_x \Delta\lambda$$

$$y = C_y \Delta L$$

where $\Delta\lambda$ and ΔL are differences in latitude and longitude.

For a spherical earth C_x and C_y are computed by

$$C_x = A \cos L_c$$

$$C_y = A$$

where L_c is the latitude of the center point,

A is the number of statute miles/degree(=69.17133901148).

For an ellipsoidal earth the values of C_x and C_y are

$$C_x = \frac{A \cos L_c}{(1-e^2 \sin^2 L_c)^{1/2}}$$

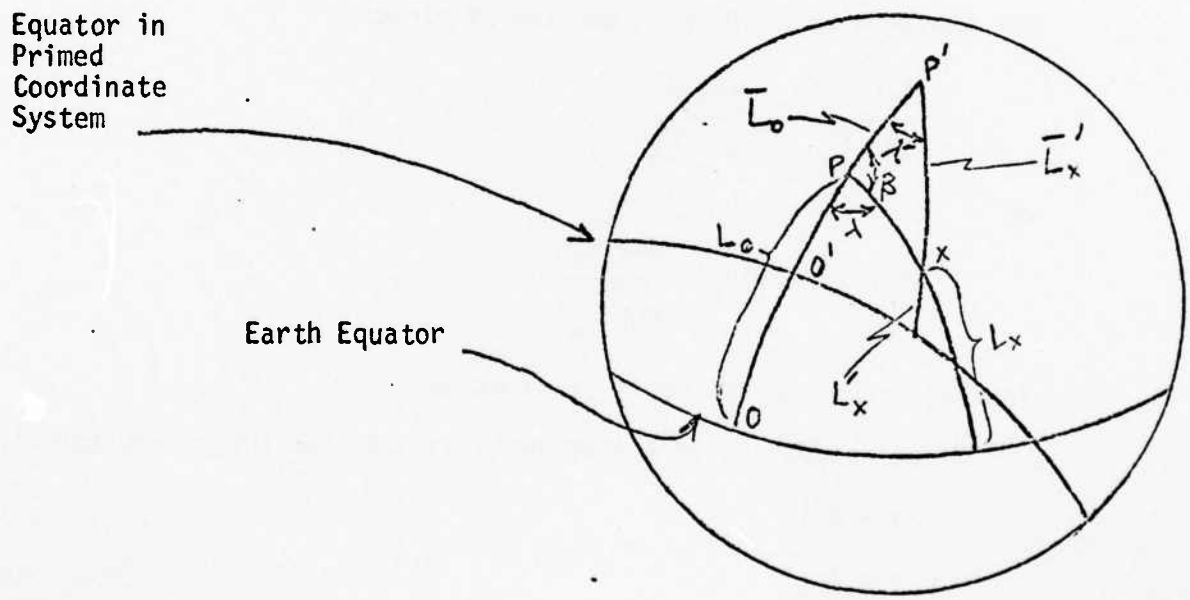
$$C_y = \frac{A(1-e^2)}{(1-e^2 \sin^2 L_c)^{3/2}}$$

COORDINATE SYSTEM ROTATION (IC = 3 or 4)

This projection rotates coordinates so that the center of projection is at the center of the coordinate system, i.e., at L_0, λ_0.

The projection then assumes a flat earth based on the new coordinate system. It gives the same errors as if the region near the earth's equator were projected with a flat earth approximation. A correction allows a stretching to minimize distance calculation errors. The equations are derived since no direct reference is known.

The following figure illustrates two coordinate systems, the unprimed natural coordinates and the primed coordinates rotated along the central meridian so the equator comes at a latitude L_0 along the central meridian. For a point x with coordinates L_x,λ in the earth coordinate system, we wish to find L_x' and λ' in the rotated coordinate system.



Let a colatitude be denoted by a bar over a symbol e.g., $\overline{L}_X = \pi/2 - L_X$.

Then

$$\sin \overline{L}_X = \cos L_X,$$

$$\cos \overline{L}_X = \sin L_X,$$

The figure illustrates, for a point x, latitudes, longitudes, and colatitud both in the earth coordinate system with pole P and the primed coordinate system with pole P'. From the law of cosines for triangle PP'x

$$\cos \overline{L}_X' = \cos L_0 \cos \overline{L}_X + \sin L_0 \sin \overline{L}_X \cos \beta$$

Since

$$\beta = 180 - \lambda,$$

$$\sin L_X' = \cos L_0 \sin L_X - \sin L_0 \cos L_X \cos \lambda.$$

For the same triangle from the law of sines

$$\frac{\sin \lambda'}{\sin \overline{L}_X} = \frac{\sin \beta}{\sin \overline{L}_X'}$$

or

$$\sin \lambda' = \sin \lambda \frac{\cos L_X}{\cos L_X'}$$

Thus L_X' and λ' can be obtained directly.

If an assumption of a flat earth is made for the primed coordinate system

$$x = A \lambda'$$

$$y = AL_X'$$

where A is the number of statute miles/degree if L_X' and λ' are in degrees.

The primed coordinate system is normally centered at $L_0 = 40^\circ$ and $\lambda_0 = 90^\circ$ West of Greenwich.

The majority of the population of the United States lives within 5° of the 40° parallel. The error in computing East-West distances is proportional to $1-\cos L_X'$. At the boundary of the 5° bank this error is 0.5 percent. The maximum error is at the tip of Florida where $L_X' = -15^\circ$. At this point the error in East-West distances is 3.5 percent.

The local errors in computing East-West distance may be lessened by dividing the x coordinate at a latitude $\bar{L}$ by cos $\bar{L}$ where $\bar{L}$ is defined as the actual latitude minus the latitude of the center of the projection. This will give correct East-West coordinates, however shapes will be distorted so that distances along lines running Northeast are lessened. To compensate for this a factor α is defined which allows stretching distances by α times the full compensation for East-West coordinates. The x coordinate stretching is then given by

$$E' = E(1 - \alpha\ (1-\cos \bar{L})).$$

To determine what the distance errors might be, imagine a displacement Δx in the easterly direction in the stretched coordinates and Δy in the northerly direction. In actual distance coordinates we have

$$\Delta y' = \Delta y,\ \Delta x' = \Delta x\ \cos \bar{L}.$$

Here locally a flat earth is assumed and only second order terms for the series expansion of cos L are retained. The displacements do not form a rectangle but a parallelogram will be vertically rotated on angle θ to the East due to the stretching of Eastward distances. We have

$$\theta = \frac{\lambda\alpha(1-\cos \bar{L})}{\bar{L}} = \frac{\alpha\lambda\bar{L}}{2}.$$

The true distance is given by

$$D^2 = \Delta y^2 + \Delta x^2 \cos^2 \bar{L}$$

and the map distance by

$$D_m^2 = \Delta y^2 + (\Delta x - \Delta y \tan \alpha)^2$$

We wish to choose a value of α to minimize $E_m = D^2 - D_m^2$ so D^2 can be best approximated by $\Delta x^2 + \Delta y^2$. Let $R = \Delta y/\Delta x$.

Then, to second order terms in cos $\bar{L}$

$$E = \Delta x^2 \bar{L}\left(-\frac{\bar{L}}{2} + R\alpha\lambda - \frac{R^2\alpha^2\lambda^2\bar{L}}{4}\right)$$

Now if $\bar{L}$ and λ are small, then E is minimized by minimizing the first two terms, in other words

$$\alpha_{opt} = \frac{\bar{L}}{2R\lambda}$$

Now moreover if Δx and Δy are identically distributed in a random distribution, then the appropriate value if L and λ are also identically distributed for R is one, and $\alpha_{opt} = \frac{1}{2}$. The stretched coordinate system uses this value of α, and extends East-West distances by

$$E = E'(1 - \frac{L^2}{4})$$

If distances which are predominantly North-South or East-West are considered those values might be changed.

ALBERS CONICAL EQUAL AREA PROJECTION (IC = 5 or 6)

The equations with two standard parallels for the Albers Conical Equal Area Projection for an ellipsoidal earth are from pp. 96-102 in the book by Deetz and Adams.[1] For a given latitude ϕ and longitude from the central meridian λ, values of polar map coordinates ρ and θ are computed. The angle θ is computed by

$$\theta = n\lambda$$

where n is a constant depending on the latitude of the standard parallels chosen (usually $29\frac{1}{2}^0$ and $45\frac{1}{2}^0$ for maps of the United States). n is computed by

$$n = \frac{a^2}{2c^2} \; \frac{\dfrac{\cos^2 \phi_1}{1-e^2 \sin^2 \phi_1} - \dfrac{\cos^2 \phi_2}{1-e^2 \sin^2 \phi_2}}{\sin \beta_2 - \sin \beta_1}$$

where a = earth's radius

e = spheroid eccentricity

ϕ_1, ϕ_2 = latitude of the standard parallels

$c^2 = a^2(1-e^2)\, F$ with

$F = 1 + 2/3\, e^2 + 3/5\, e^4 + 4/7\, e^6$

β_1, β_2 are authalic latitudes.

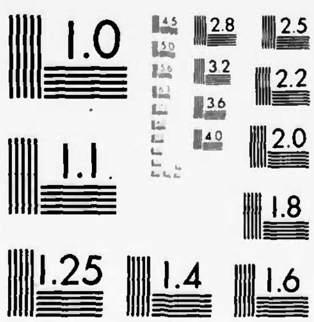
[1]Charles H. Deetz and Oscar S. Adams, "Elements of Map Projection," U.S. Coast and Geodetic Survey, Special Publication No. 68, U.S. Government Printing Office, Washington, D. C., 1945

AD-A039 819 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA PROGRAM --ETC F/G 15/4
DOCUMENTATION OF CURRENT IDA COMPUTER MATERIAL DEVELOPED FOR DC--ETC(U)
JAN 77 L A SCHMIDT DCPA01-76-C-0213

UNCLASSIFIED IDA/HQ-77-19226 NL

2 OF 3
ADA039 819

039



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

Authalic latitudes are computed from actual latitudes by the equation

$$\sin \beta = \sin \phi \left(\frac{1 + \frac{2}{3} e^2 \sin^2 \phi + \frac{3}{5} e^4 \sin^4 \phi + \frac{4}{7} e^6 \sin^6 \phi}{1 + \frac{2}{3} e^2 + \frac{3}{5} e^4 + \frac{4}{7} e^6} \right)$$

The radius ρ is computed by

$$\rho^2 = \rho_2{}^2 + \frac{4c^2}{n} \frac{\sin \beta_2 - \sin \beta}{2}$$

where ρ_2 is the radius for one of the standard parallels computed by the formula[2]

$$\rho_2 = \frac{a \cos \phi_2}{n(1-e^2 \sin^2 \phi_2)^{1/2}}$$

Given values of ρ and θ rectangular coordinates x and y are computed by

$$x = \rho \sin \theta$$

$$y = \rho_0 - \rho \cos \theta$$

Where ρ is found for the center of the rectangular coordinate system (for maps of the U.S. this is usually taken as 39° latitude and 96° longitude). The units of x and y are the same as the units of earth's radius, i.e., statute miles.

For a spherical earth the equations used are simply found by setting $e = 0$ in the above equations. One consequence is that the authalic latitude β, becomes equal to the actual latitude ϕ.

<u>MERCATOR PROJECTION</u> (IC = 7 or 8)

The equations for a Mercator Projection are from Deetz and Adams, p. 114.[2] We have

$$x = a\lambda$$

$$y = a \log_e (1/\tan(z/2))$$

where $z/2$ is defined by

$$\tan z/2 = \tan p/2 \left(\frac{1 + e \cos p}{1 - e \cos p}\right)^{e/2}$$

with $p = \pi/2 - \phi$.

LAMBERT CONFORMAL CONIC PROJECTION WITH TWO STANDARD PARALLELS (IC = 9)

The Lambert Conformal Conic is discussed briefly in Deetz and Adams Special Publication No. 68. The basic equations except for the series expansions are taken from Special Publication No. 52[2].

The transformation preserves right angles and is exact on two standard parallels.

A variable z is defined by the equation

$$\tan z/2 = \tan p/2 \left(\frac{1 + e \cos p}{1 - e \cos p}\right)^{e/2}$$

where

e is the spheroid eccentricity

p is the colatitude $= \pi/2 - \phi$.

The term $\left(\frac{1 + e \cos p}{1 - e \cos p}\right)^{e/2}$ is a correction term with value near 1.

For any value of ϕ a radius r is computed by

$$r = K(\tan z/2)^{\ell}$$

Where ℓ is computed from

$$\ell \log \left(\frac{\tan z_1/2}{\tan z_2/2}\right) = \log \frac{(1-e^2 \sin^2 \phi_2)^{\frac{1}{2}} \cos \phi_1}{(1-e^2 \sin^2 \phi_1)^{\frac{1}{2}} \cos \phi_2}$$

and K is computed from

$$K = \frac{a \cos \phi_1}{(1-e^2 \sin^2 \phi_1)^{1/2} \cdot \ell(\tan z_1/2)^{\ell}}$$

[2] Oscar S. Adams, "Lambert Projection Tables for the United States," U. S. Coast and Geodetic Survey, Special Publication #52, U. S. Government Printing Office Washington, D.C., 1918

Here ϕ_1 and ϕ_2 are the latitudes of the standard parallels, usually 45^o and 39^o for maps of the United States. K and ℓ are computed in the initialization. A radius r_0 is computed for the latitude of the center of the map (39^o) for U.S. maps. Then rectangular coordinates x and y are computed by

$$x = r \sin \ell\lambda$$

$$y = r_0 - r \cos \ell\lambda$$

where λ is the longitude from the central meridian.

THE TRANSVERSE MERCATOR PROJECTION (IC = 10)

The transverse mercator projection is basically a mercator projection rotated 90^o so one meridian is presented at true length, instead of the equator. The user is free to choose the center meridian by the value of FLONC, as opposed to the universal transverse mercator projection when the center value forced by the zone system. The formulas used here from Thomas[3] are suggested for bands running 2^o from the central meridian, as opposed to more extensive formulas for 5 to 6^o bands. For most map projection purposes however, these formulas should be adequate for larger band, and are faster to use than the higher order calculation. We have, in Thomas's notations

$$x/N = \frac{\Delta\lambda}{\rho} \cos\phi \frac{\Delta\lambda^3 \cos^3\phi}{6\rho^3} (1-t^2+n^2) + \frac{\Delta\lambda^5 \cos^5\phi}{120\rho^5} (5-18t^2+t^4)$$

$$y/N = \frac{S\phi}{N} + \frac{\Delta\lambda^2}{2\rho^2} \sin\phi \cos\phi + \frac{\Delta\lambda^4}{24\rho^4} (\sin\phi \cos^3\phi)(5-t^2)$$

Where ϕ = latitude $\Delta\lambda$ = longitude from central meridian, degrees

ρ = cosec 1" (in the implementation since λ is in radians this conversion is not used)

$t = \tan\phi$

$$N^2 = \frac{\varepsilon^2}{1-\varepsilon^2} \cos^2\phi$$

$$N = a/(1-\varepsilon^2 \sin^2\phi)^{1/2}$$

S_ϕ = meridian are from the equator to latitude ϕ, (see Polyconic projection for a description of the calculation of S_ϕ)

[3]Paul D. Thomas, "Conformal Projections in Geology and Cartography", Special publication #251, U.S. Department of Commerce, Coast Geodetic Survey, U. S. Government Printing Office, 1952, p. 4.

UNIVERSAL TRANSVERSE MERCATOR PROJECTION (IC = 11)

The universal transverse mercator projection outputs distance in meters on grids defined for zones covering the earth. There are 60 zones beginning with #1 covering from 180° to 174° West Longitude and sequentially increases to number 60 from 174° to 180° East Longitude. Thus for example zone 17 in the United States extends from Longitude 84° to 78°. In each zone distances North of the equator in meters, and distances in meters East of the central zone meridian (eq. 81° in zone 17), increased by 500,000 meters to keep all Eastings positive, are output. In the projection the output distance Y and X are in statute miles, and the control parameter KC is restricted to 1. The Northing and Easting values in meters are given by the variables RHO and THETA in common block /CARTOG/.

The implementation is based on the Army Technical Manual defining the UTM grid[4], and appears accurate at least to the nearest meter. The zone control is governed by the parameter ROLATR on an initialization call. If ROLATR is equal to 0 the zone selected is that one containing FLONC. If ROLATR is not zero the zone selected is the value of ROLATR (for a value of ROLATR less than 0 or greater than 60 an error stop is given). The spheroid selected is determined by the value of HILATR in the initialization call. A value of 0 or 1 selects the Clark 1866 spheroid. A value of 2, 3, 4 or 5 selects the Clark 1880, International, Everest, or Bessel spheroids respectively. A map in the Technical Manual indicates the spheroid to be used for different world areas. Roughly they are Clark 1866 for North America and the Phillipine Islands; Clark 1880 for Africa South of the Sahara Desert; Everest for India, Pakistan, Afghanistan and Indochina; Bessel for Indonesia, Japan, Korea and Manchuria; and International elsewhere.

The calculation of Northing and Easting in the notation of the manual is given by

$$N = (I) + (II)p^2 + (III)p^4 + A_6$$

$$E = (IV)p + (V)p^3 + B_5$$

(For the Southern Hemisphere a false Northing of 10,000,000 meters is added.)

where

$p = 0.0001\,\Delta\lambda$ with $\Delta\lambda$ the longitude from central meridian in seconds of arc

$$(I) = S_\phi K_0$$

[4]"The Universal Transverse Mercator Guide", Department of the Army Technical Manual TM 5-241-8, Headquarters, Department of the Army, July 1958.

$$(II) = \frac{\nu \sin\phi \cos\phi \sin^2 1'' \, k_0 \cdot 10^6}{2}$$

$$(III) = \frac{\sin^4 1'' \, \nu \sin\phi \cos^3\phi}{24} \, (5-\tan^2\phi + 9e'^2 \cos^2\phi) \, k_0 \cdot 10^{16}$$

$$(IV) = \nu \cos\phi \sin 1'' \cdot k_0 \cdot 10^4$$

$$(V) = \frac{\sin^3 1'' \, \nu \cos^3\phi}{6} \, (1-\tan^2\phi + e'^2 \cos^2\phi) \, k_0 \cdot 10^{12}$$

$$A_6 = \frac{p^6 \sin^6 1'' \cdot \nu \sin\phi \cos^5\phi}{720} \, (61-58 \tan^2\phi + \tan^4\phi$$

$$+ 270e'^2 \cos^2\phi - 330e'^2 \sin^2\phi) \, x k_0 \times 10^{24}$$

$$B_5 = \frac{p^6 \sin^5 1'' \, \nu \cos^5\phi}{120} \, (5-18 \tan^2\phi + \tan^4\phi + 14e'^2 \cos^2\phi$$

$$- 58e'^2 \sin^2\phi) \cdot k_0 \cdot 10^{20}$$

with $S\phi$ = meridional distances on the spheroid from the equator (see polyconic projection).

k_0 = central scale factor, an arbitrary factor to reduce the maximum scale distortion = 0.9996

$\sin 1''$ = conversion factor = $4.848136811 \times 10^{-6}$

ϕ = latitude

$\nu = a/(1-e^2\sin^2\phi)^{1/2}$

$e'^2 = e^2/1-e^2$

The eccentricity is computed from the flattening, f, which is used to define the spheroid by

$$e^2 = \frac{1}{f}\left(2-\frac{1}{f}\right)$$

POLYCONIC PROJECTION (IC = 12)

The equations for the Polyconic Projection are from Oscar S. Adams, "General Theory of Polyconic Projection", U. S. Coast & Geodetic Survey Special Publication No. 57, U.S. Government Printing Office, Washington, D. C., 1934. The equations, except for the integration, are from the section on "Ordinary, or American, Polyconic Projection" pp. 143-152. A spheroid is assumed as the earth shape.

Let ϕ be latitude, λ longitude from the central meridian on earth. On a map let s be the distances up from the origin to a point s, θ the angle on the map, and ρ the length of a vector from s to the point to be plotted.

Then:

$$\rho = \frac{a \cot \phi}{(1-e^2 \sin^2 \phi)^{1/2}}$$

$$\theta = \lambda \sin \phi$$

$$s = a\,(1-e^2) \int_0^\phi \frac{d\,\phi}{(1-e^2 \sin^2 \phi)^{3/2}} + \frac{a \cot \phi}{(1-e^2 \sin^2 \phi)^{1/2}}$$

The first term is the true meridional distance from the equator along the spheroid also used in some other projections. This integral is calculated by expanding the denominator in a series.

Thus

$$\int_0^\phi \frac{d\,\phi}{(1-e^2 \sin^2 \phi)^{3/2}} = \int_0^\phi (1 + \frac{3}{2} \varepsilon^2 \sin^2 \phi + \frac{3}{2} \cdot \frac{5}{4} \varepsilon^4 \sin^4 \phi$$

$$+ \frac{3}{2} \cdot \frac{5}{4} \cdot \frac{4}{6} \cdots\cdots \frac{2\,n+1}{2n}\, \varepsilon^{2n} \sin^{2n} \phi)\, d\,\phi$$

Now define: $T_1 = \int_0^\phi \sin^2 \phi = \frac{\phi}{2} - \frac{\sin\phi \cos \phi}{2}$

and

$$T_n = \int_0^\phi \sin^{2n} \phi d\,\phi = -\frac{\sin^{2n-1} \phi \cos \phi}{2^n} + \frac{2\,n-1}{2n} \int_0^\phi \sin^{2n-2}\phi\, d\,\phi$$

Then

$$T_n = -\frac{\sin^{2n-1}\phi \cos\phi}{2n} + \frac{2^{n-1}}{2n} T_{n-1}$$

Call

$$\alpha = \frac{\sin\phi \cos\phi}{2}$$

so

$$T_{n-1} = -\frac{\sin^{2n-2}\phi\ \alpha}{n} + \frac{2\ n-1}{2n} T_{n-1}$$

Call

$$F_1 = \frac{3}{2}\varepsilon^2$$

$$F_j = \varepsilon^2 + \frac{2j+1}{2j} F_{j-1}$$

Then

$$\int_0^\phi \frac{d\ \ \phi}{(1-\varepsilon^2\ \sin^2)^{3/2}} = \phi + F_1 T_1 + F_2 T_2 + F_3 T_3 + \ldots$$

A recursion formula is readily coded to calculate F_j and T_j from F_{j-1} and T_{j-1}. Carrying the series to $j = 4$ yields adequate accuracy for this purpose.

The x and y coordinates are those formed from

$$x = \rho \sin\theta$$

$$y = s - \rho \cos\theta$$

TRAPEZOIDAL PROJECTION (IC = 13)

The trapazoidal projection is a simple projection which offers an improvement over the flat earth projection by accounting for the convergence of meridian at the map central latitude. It assumes the meridians are straight lines.

Let L_0, λ_0 of the latitude and longitude for the center of the projection and L and λ be the latitude and longitude of any point. Then

$$X = (\lambda_0 - \lambda) \cdot \cos L_0 - (L-L_0) \sin L_0$$

$$Y = L-L_0.$$

X and Y are multiplied by the appropriate conversion factors to yield statute miles. The subroutine must be initialized to define FLATC and FLONC.

STEROGRAPHIC HORIZON PROJECTION ON SPHERICAL EARTH (IC = 14)

In the sterographic projection, it is assumed the earth is projected on a plane tangent to the globe as viewed from a point on the earth surface opposed to the point of contact with the plane and the earth. In this sterographic horizon projection the point of contact is associated with the latitude and longitude L_0, λ_0. If JC=3 these are given by FLATC and FLONC, if JC=2 by 40^o and 90^o. The equations are given by Thomas[5]. If L is a latitude and λ longitude from the center of projection

$$X = a \sin \lambda \cos L/(1 + \sin L \sin L_0 + \cos L \cos L_0 \cos \lambda)$$

$$Y = a (\sin L \cos L_0 - \sin L_0 \cos L \cos \lambda) / (1 + \sin L \sin L_0 + \cos L \cos L_0 \cos \lambda)$$

OUTPUT FORMATS

The outputs for KC = 1 are directly the output for each type of projection in statute miles centered at FLATP, FLONP. If X', Y' are the final subroutine outputs and X,Y the results of the projection calculation, then for KC = 1 X' = X and Y' = Y. For other values of KC the following transformations are performed.

(KC = 2)Grid coordinate in statute miles

$$X' = (X + XGDST)/50$$

$$Y' = (Y + YGDST)/50$$

[5]Thomas, Paul D., "Conformal Projection in Geodesy and Cartography", U. S. Dept. of Commerce, Coast and Geodetic Survey, Special Publication #251, U. S. Government Printing Office, Washington, D.C., 1952, p. 15.

for

$JC = 2$ $XGDST = 2000$, $YGDST = 1000$

$JC = 3$ $XGDST = XGDSTR$, $YGDST = YGDSTR$

(KC = 3)Map coordinates in inches

$$X' = X \cdot SMIIN \cdot SCLEW - XMPOF$$

$$Y' = Y \cdot SMIIN \cdot SCLNS - YMPOF$$

where

SMIIN = 63360 inches/statute mile

SCLEW = STRCHA/SCALE

SCLNS = STRCHU/SCALE

SCALE is input in block common /CARTOG/

if JC = 2 XMPOF = YMPOF = 0,

STRCHA = STRCHU = 1.

if JC = 3 XMPOF = XMPOFR,

XMPOF = YMPOFR,

STRCHA = STCHAR,

STRCHU = STCHUR,

(KC = 4)Rectangular coordinates in nautical miles

$$X' = SMINM \cdot X$$

$$Y' = SMINM \cdot Y$$

where

SMINM = 0.867383901158 Nautical miles/Statute mile,

(KC = 5)Grid coordinates in nautical miles

$$X' = (X + XGDST)\ SMINM/60$$

$$Y' = (Y + YGDST)\ SMINM/60$$

where

XGDST, YGDST is the same as when KC = 2

(KC = 6) Offset map center - output in inches.

$$X' = (X - XCENT) \cdot SMIIN \cdot SCLEW - XMPOF$$

$$Y' = (Y - YCENT) \cdot SMIIN \cdot SCLNS - YMPOF$$

where

SCLEW, SCLNS, XMPOF, YMPOF are as defined for KC = 3

XCENT, YCENT are obtained by a preliminary call calculating a data point using the value CRMPLA for FLATP, and CRMPLO for FLONP. This puts the center of the map at this latitude and longitude.

```
      SUBROUTINE PROJCT
C     NEVUNS STANDARD
C     LAST REVISED MARCH 27,1973

C     THIS SUBROUTINE PROJECTS FROM LATITUDE LONGITUDE COORDINATES TO
C     AN X Y SET OF RECTANGULAR COORDINATES
C     SEVERAL OPTIONS,CONTROLLED BY PARAMETER IC ARE --
C ... IC =  1 --- FLAT SPHERICAL EARTH
C ... IC =  2 --- FLAT ELLIPSOIDAL EARTH
C ... IC =  3 --- ROTATE COORDINATE SYSTEM BEFORE PROJECTION
C ... IC =  4 --- AS 3 BUT ALSO STRETCH TO EQUALIZE ERRORS
C ... IC =  5 --- ALBERS EQUAL AREA PROJECTION ON ELLIPSOIDAL EARTH
C ... IC =  6 --- ALBERS EQUAL AREA PROJECTION ON SPHERICAL EARTH
C ... IC =  7 --- MERCATOR PROJECTION ON ELLIPSOIDAL EARTH
C ... IC =  8 --- MERCATOR PROJECTION ON SPHERICAL EARTH
C ... IC =  9 --- LAMBERT CONFORMAL CONIC PROJECTION, ELLIPSOIDAL EARTH
C ... IC = 10 --- TRANSVERSE MERCATOR PROJECTION ON ELLIPSOIDAL EARTH
C ... IC = 11 --- UNIVERSAL TRANSVERSE MERCATOR PROJECTION
C ... IC = 12 --- POLYCONIC PROJECTION ON ELLIPSOIDAL EARTH
C ... IC = 13 --- TRAPAZOIDAL PROJECTION ON SPHERICAL EARTH
C ... IC = 14 --- STEROGRAPHIC HORIZON PROJECTION ON SPHERICAL EARTH
C     THE PARAMETER JC CONTROLS INITIALIZATION. TYPICALLY JC = 1
C     FINDS COORDINATES FOR A POINT, JC = 2 INTITIATES WITH STANDARD
C     PARAMETERS, JC = 3 INITIATES WITH USER SUPPLIED PARAMETERS.
C     THE PARAMETER KC CONTROLLS OUTPUT TYPE TYPICALLY THE OUTPUT IS
C ... KC =  1 --- OUTPUT IN RECTANGULAR COORDINATES IN STATUTE MILES
C ... KC =  2 --- OUTPUT IN GRID COORDINATES IN STATUTE MILES
C ... KC =  3 --- OUTPUT IN INCHES FOR DRAWING MAPS
C ... KC =  4 --- OUTPUT IN RECTANGULAR COORDINATES IN NAUTICAL MILES
C ... KC =  5 --- OUTPUT IN GRID COORDINATES IN NAUTICAL MILES
C ... KC =  6 --- OUTPUT FOR MAPS OF ENLARGED SCALE LOCATED AWAY FROM
C                   THE COORDINATE SYSTEM CENTER

      DIMENSION CLA(5),FLT(5)
      COMMON/CARTOG/IC,JC,KC,FLATP,FLONP,Y,X,RHO,THETA, FLATC,FLONC,
     1SCALE, STCHUR,STCHAR,HILATR,ROLATR,XGDSTR,YGDSTR,XMPOFR,YMPOFR,
     2 CRMPLA,CRMPLO
      DATA TORAD,RETIN,SMIIN,SMINM/  0.0174532925199, 39.37, 63360.,
     1 0.867383901158/
      DATA RETSM/1609.3472186944 /
      DATA CLARK,ECC,ECCSQ,STMIO,AIMES,ECCRTO,ECCH/
     1 3963.225788636, 8.22718542578E-2,    6.768658002191E-3,
     2 69.1713390I148,3936.1044794, 6.8147849509E-3,4.1135927126E-2/
      DATA XKO,SINS/ 0.9996 ,4.84813681IE-6/
      DATA HILATO,ROLATO,CENLOD,CENLAD /45.5,29.5,96.,40./
      DATA HILALO,ROLALO,CENLLD,CELLOO/ 45.,33.,39.,96./
      DATA FLATM, FLONM/40.,90./
      DATA CTMLAT,CTMLON/ 39.,90./
      DATA FLATPY,FLONPY/ 40.,90./
```

```
              DATA CLTMER,CLNMER/ 0.,90./
              DATA XGDSTD,YGDSTO,XMPOFD,YMPOFD/2000.,1000.,30.,15./
              DATA OTRCHU,OTRCHA/1.,1./
              DATA CLA(1),CLA(2),CLA(3),CLA(4),CLA(5)/
             16378206.4,6378249.145,6378388.,6377276.3452,6377397.155/
              DATA FLT(1),FLT(2), FLT(3), FLT(4), FLT(5)/
             1 294.978698, 293.465, 297., 300.8017,299.152813/


              IF (JC . EQ. 1) GO TO 200


        C     INITIALIZE FOR ALL CODE UP TO STATEMENT 200

              IF (IC .GT. 2) GO TO 50

        C     PROJECT LOCALLY UPON A FLAT EARTH
        C     IC = 1 USE SPHERICAL FLAT EARTH,IC = 2 USE ELLIPSOIDAL EARTH
        C     FOR STRETCH FACTORS
        C     JC = 1 COMPUTE A POINT, JC = 2 INITIALIZE WITH PRESTORED DATA
        C     JC = 3 INITIALIZE WITH DATA IN COMMON BLOCK /CARTOG/
              IF (IC . EQ.1) GO TO 13

        C     FLAT ELLIPSOIDAL EARTH
              SINLT = SIN(FLATC*TORAD)
              SINLTS = SINLT*SINLT
              COSLT = SQRT(1.-SINLTS)
              TMP = SQRT(1.-ECCSQ*SINLTS)
              FCTN =(1. - ECCSQ)/(TMP*TMP*TMP)
              FCTE = COSLT/TMP
              GO TO 14
        13    CONTINUE

        C     FLAT SPHERICAL EARTH
              FCTN = 1.
              FCTE = COS(FLATC*TORAD)
        14    CONTINUE
              CONNO = FCTN*STMID
              CONEA  = FCTE*STMID
              GO TO 16
        50    CONTINUE
              IF(IC .GT. 4) GO TO 20



        C     ROTATE COORDINATE SYSTEM SO CENTERED AT PT LO
        C     THEN USE FLAT EARTH OR FLAT STRETCHED EARTH
              IF(JC .EQ. 3) GO TO 52
        C     JC = 2 USE STANDARD CENTER AT LAT = 40, LONG = 90
              SINLO = SIN(FLATM*TORAD)
              COSLO = COS(FLATM*TORAD)
              FLONUS = FLONM
              FLATUS = FLATM
              GO TO 16
        C     JC = 3 USE FLATC FLONC AS CENTER
```

```
   52   CONTINUE
        SINLO = SIN(FLATC*TORAD)
        COSLO = COS(FLATC*TORAD)
        FLONUS = FLONC
        FLATUS = FLATC
        GO TO 16

   20   CONTINUE
        IF ( IC .GT. 6) GO TO 70


  C     ALBERS EQUAL AREA PROJECTION
  C ... IC =  5 --- ALBERS EQUAL AREA PROJECTION ON ELLIPSOIDAL EARTH
  C ... IC =  6 --- ALBERS EQUAL AREA PROJECTION ON SPHERICAL EARTH

        IF (JC .NE. 3) GO TO 31

  C     JC = 3 USER MUST SUPPLY VALUES FOR STANDARD PARALLELS AND CENTER
  C     LONGITUDE AND LATITUDE
        HILAT = HILATR
        ROLAT = ROLATR
        CENLON = FLONC
        CENLAT = FLATC
        GO TO 38
   31   CONTINUE
        HILAT = HILATD
        ROLAT = ROLATD
        CENLON = CENLOD
        CENLAT = CENLAD
   38   CONTINUE

        ASQ = CLARK*CLARK
        COSHI = COS(HILAT*TORAD)
        COSHIS = COSHI*COSHI
        COSLO = COS(ROLAT*TORAD)
        COSLOS = COSLO*COSLO
        SINHIS = 1. - COSHIS
        SINLOS = 1. - COSLOS
        SINHI = SQRT(SINHIS)
        SINLO = SQRT(SINLOS)
        IF( IC .EQ. 6) GO TO 21

  C     ELLIPSOIDAL EARTH CLARK STD SPHEROID OF 1866
        C1 = 0.66666666667*ECCSQ
        C2 = 0.6*ECCSQ*ECCSQ
        C3 = 0.57142857*ECCSQ*ECCSQ*ECCSQ
        OIV = 1. + C1 + C2 + C3

        SI = SINHI
        SIS = SINHIS
        ISW = 1
   32   CONTINUE
        SB = SI*(1. + C1*SIS + C2*SIS*SIS + C3*SIS*SIS*SIS)/OIV
        GO TO ( 34,35),ISW
```

```
      34    CONTINUE
            SINBHI = SB
            SI = SINLO
            SIS = SINLOS
            ISW = 2
            GO TO 32
      35    CONTINUE
            SINBLO = SB
            ASSCS = 1./((1. - ECCSQ)*DIV)
            XN = 0.5*ASSCS*(COSLOS/(1. - ECCSQ*SINLOS) -COSHIS/(1.- ECCSQ*
           1 SINHIS))/(SINBHI - SINBLO)
            RHOHIS = ASQ*COSHIS/(XN*XN*(1.- ECCSQ*SINHIS))
            FACT = 2.*ASQ/(ASSCS*XN)
            GO TO 39
      C     SPHERICAL EARTH
      21    CONTINUE
            XN = 0.5*(COSLOS - COSHIS)/(SINHI - SINLO)
            RHOHIS = ASQ*COSHIS/(XN*XN)
            FACT = 2.*ASQ/XN
            SINBHI = SINHI
      39    CONTINUE
            FLTPST = FLATP
            FLATP = CENLAT
            GO TO 220
      41    CONTINUE
            FLATP = FLTPST
            RHOC = RHO
            GO TO 16
      70    CONTINUE
            IF ( IC.GT. 8) GO TO 120
      C ..., IC =  7 --- MERCATOR PROJECTION ON ELLIPSOIDAL EARTH
      C ..., IC =  8 --- MERCATOR PROJECTION ON SPHERICAL EARTH
      C     INITIALIZE FOR CENTER POINT ON MAP
            FLONST = FLONP
            FLATST = FLATP
            IF(JC .NE. 3) GO TO 125
            FLONP = FLONC
            FLATP = FLATC
            GO TO 126
      125   CONTINUE
            FLATP =  CLTMER
            FLONP = CLNMER
      126   CONTINUE
            XO = 0.
            YO = 0.
            GO TO 230
      131   CONTINUE
            XO = X
            YO = Y
            FLATP = FLATST
            FLONP = FLONST
```

```
          GO TO 16
    120   CONTINUE
          IF(IC .GT. 9) GO TO 130

C         LAMBERT CONFORMAL CONIC WITH TWO STANDARD PARRALLELS
C         ASSUME ELLIPSOIDAL EARTH
          IF (JC .NE.3) GO TO 121
          HILTA = HILATR
          ROLTA = ROLATR
          CENLA = FLATC
          CENLN = FLONC
          GO TO 122
    121   CONTINUE
          HILTA = HILALD
          ROLTA = ROLALD
          CENLA = CENLLO
          CENLN = CELLOD
    122   CONTINUE
          SINLO = SIN(ROLTA*TORAD)
          COSLO = COS(ROLTA*TORAD)
          SINHI = SIN( HILTA*TORAD)
          COSHI = COS(HILTA*TORAD)
          RTHI= SQRT(1. - ECCSQ*SINHI*SINHI)
          RTLO= SQRT(1. - ECCSQ*SINLO*SINLO)
          FACLA = RTLO*COSHI/(RTHI* COSLO)
          PLO = 1.57079632 - ROLTA*TORAD
          PHI = 1.57079632 - HILTA*TORAD
          COSPHI = SINHI
          COSPLO = SINLO
          BRHI = ((1. + ECC*COSPHI)/( 1.- ECC*COSPHI))**( ECC/2.)
          BRLO = ((1. + ECC*COSPLO)/( 1.- ECC*COSPLO))**( ECC/2.)
          TZHI = TAN (PHI/2.)*BRHI
          TZLO = TAN(PLO/2.)*BRLO
          XL = ALOG10(FACLA)/ALOG10(TZHI/TZLO)
          XK = CLARK*COSHI/(RTHI*XL*TZHI**XL)
C         INITIALIZE FOR CENTER POINT ON MAP
          FLONST = FLONP
          FLATST = FLATP
          FLONP = CENLN
          FLATP = CENLA
          RHOO = 0.
          GO TO 240
    124   CONTINUE
          RHOO = RHO
          FLATP = FLATST
          FLONP = FLONST
          GO TO 16
    130   CONTINUE
          IF( IC.GT. 10) GO TO 150

C ...     IC = 10 --- TRANSVERSE MERCATOR PROJECTION CENTERED ON FLONC
          IF(JC .NE.3) GO TO 132
```

```
      CENLA = FLATC
      CENLO = FLONC
      GO TO 133
  132 CONTINUE
      CENLA = CTMLAT
      CENLO = CTMLON
  133 CONTINUE
      FLATST = FLATP
      FLONST = FLONP
      FLATP = CENLA
      FLONP = CENLO
      GO TO 250
  134 CONTINUE
      YCNT = Y
      FLATP = FLATST
      FLONP = FLONST
      GO TO 16
  150 CONTINUE
      IF( IC.GT. 11) GO TO 170

C ... IC = 11 --- UNIVERSAL TRANSVERSE MERCATOR PROJECTION
C     USE CLARK 1866 FOR NORTH AMERICA AND PHILLIPINE ISLANDS,
C     CLARK 1880 FOR AFRICA SOUTH OF THE SAHARA, EVEREST FOR INDIA,
C     PAKISTAN,AFGHANISTAN,AND INDOCHINA, BESSEL FOR INDONESIA,JAPAN,
C     KOREA, AND MANCHURIA, AND INTERNATIONAL ELSEWHERE.
C     THE VALUE OF HILATR IS USED TO  DEFINE THE SHEROID IN THE
C     FOLLOWING INCREASING ORDER.
C                 CLARK 1866, CLARK 1880, INTERNATIONAL, EVEREST, AND
C     BESSEL SPHEROIDS
C     CLEARLY FOR THE U.S. SET HILATR = 1.
      IF(HILATR .LT. 0. .OR. HILATR .GT. 5.0001) STOP 436
      IND = HILATR + 0.0000001
      IF(HILATR .EQ. 0.) IND = 1
      CLARKU = CLA(IND)
      FLTNU = 1./FLT(IND)
      ECCSU = FLTNU*(2. - FLTNU)
      ECCPS = ECCSU/(1. - ECCSU)
      ECCPF = ECCPS*ECCPS
      SINSS = SINS*SINS
      SINSC = SINS*SINSS
      SINSF = SINSS*SINSS
      SINSFT = SINSF*SINS
      SINSSX = SINSS*SINSF
C     ZONE NUMBERING STARTS WITH 1 AT 174 TO 180 W, 2 AT 174 TO 168 W
C     TO 60 AT 174 TO 180E. GRIDC IN THE CENTER OF THE ZONES.
C     ZONE NUMBER IS SPECIFIED BY THE VALUE OF ROLATR
C     IN CONUS FOR GIVEN ZONES, MIN LONGITUDE IS --
C      Z19-66, Z18-72, Z17 -78,Z16 - 84, Z15- 90, Z14 - 96, Z13 -102,
C     Z12 - 108, Z11 114, Z10 - 120. THUS FOR EX ZONE 16 COVERS 84
C     TO 90 DEGREES LONGITUDE.
      IF(ROLATR .LT. 0. .OR. ROLATR .GT. 60.) STOP 437
      IF(ROLATR .NE. 0.) GO TO 154
      IF(FLONC .LT. 0.)GO TO 155
      ITMP = FLONC/6.
```

```
      TMP = ITMP
      ROLATR = 30. - TMP
      GO TO 154
155   CONTINUE
      ITMP = ABS(FLONC)/6.
      TMP = ITMP
      ROLATR = 31. + TMP
154   CONTINUE
      INZN = ROLATR + 0.0000001
      ZONEN = INZN
      IF(ZONEN .GT. 30.) GO TO 156
      GRIDC = 180. - (ZONEN - 1.)*6. - 3.
      GO TO 157
156   CONTINUE
      GRIDC = -1.*( (ZONEN - 31.)*6. + 3.)
157   CONTINUE
      GO TO 16
170   CONTINUE
      IF (IC .GT. 12) GO TO 180

C ... IC = 12 --- POLYCONIC PROJECTION ON ELLIPSOIDAL EARTH
      FLATPS = FLATP
      FLONPS = FLONP
      IF(JC .NE.3) GO TO 172
      CENLA = FLATC
      CENLO = FLONC
      GO TO 173
172   CONTINUE
      CENLA = FLATPY
      CENLO = FLONPY
173   CONTINUE
      FLATP = CENLA
      FLONP = CENLO
      GO TO 290
171   CONTINUE
      FLONP = FLONPS
      FLATP = FLATPS
      YPC = Y
      GO TO 16
180   CONTINUE
      IF(IC .GT. 13) GO TO 190

C ... IC = 13 --- TRAPAZOIDAL PROJECTION ON SPHERICAL EARTH
      SINLC = SIN(FLATC*TORAD)*STMID
      COSLC = COS(FLATC*TORAD)*STMID
      GO TO 16
190   CONTINUE

C ... IC = 14 --- STEROGRAPHIC HORIZON PROJECTION ON SPHERICAL EARTH
      IF (JC .NE. 3) GO TO 192
      CENLA = FLATC
      CENLO = FLONC
```

```
      GO TO 193
  192 CONTINUE
      CENLA =  FLATM
      CENLO = FLONM
  193 CONTINUE
      SINLO = SIN(CENLA*TORAD)
      COSLO = COS(CENLA*TORAD)
      TCLRK = 2.*CLARK

C     THIS SECTION DOES INITIALIZATION OF THE FINAL PRINTOUT
   16 CONTINUE
      IF(KC .NE. 1) GO TO 12
C     OUTPUT IN STATUTE MILES
      RETURN
   12 CONTINUE
      IF (KC .GT.2) GO TO 17

C     KC = 2
      IF (JC . EQ.3) GO TO 22
      XGDST = XGDSTD
      YGDST = YGDSTD
      GO TO 23
   22 CONTINUE
      XGDST = XGDSTR
      YGDST = YGDSTR
   23 CONTINUE
      RETURN
   17 CONTINUE
      IF (KC .GT.3) GO TO 18

C     KC = 3
      IF (JC . EQ.3) GO TO 24
      XMPOF = XMPOFD
      YMPOF = YMPOFD
      STRCHU = DTRCHU
      STRCHA = DTRCHA
      GO TO 25
   24 CONTINUE
      XMPOF = XMPOFR
      YMPOF = YMPOFR
      STRCHA = STCHAR
      STRCHU = STCHUR
   25 CONTINUE
      SCLNS = STRCHU/SCALE
      SCLEW = STRCHA/SCALE
      RETURN
C     NO INITIALIZATION NEEDED FOR KC = 4
   18 CONTINUE
      IF (KC .GT.5) GO TO 19

C     KC = 5
      IF (JC . EQ.3) GO TO 26
```

```
      YGDST = YGDSTD
      XGDST = XGDSTD
      GO TO 27
26    CONTINUE
      YGDST = YGDSTR
      XGDST = XGDSTR
27    CONTINUE
      RETURN
19    CONTINUE

C     KC = 6
      IF (JC . EQ.3) GO TO 28
      XMPOF = XMPOFD
      YMPOF = YMPOFD
      STRCHA = DTRCHA
      STRCHU = DTRCHU
      GO TO 29
28    CONTINUE
      STRCHU = STCHUR
      STRCHA = STCHAR
      XMPOF = XMPOFR
      YMPOF = YMPOFR
29    CONTINUE
      SCLNS = STRCHU/SCALE
      SCLEW = STRCHA/SCALE
      FLATST = FLATP
      FLONST = FLONP
      JCST = JC
      JC = 1
      KC6SW = 2
      FLATP = CRMPLA
      FLONP = CRMPLO
      GO TO 200
C     DO PRELIMINARY RUNTHROUGH TO GET XCENT,YCENT
15    CONTINUE
      KC6SW = 1
      FLATP = FLATST
      FLONP = FLONST
      JC = JCST
      XCENT = X
      YCENT = Y
      RETURN


200   CONTINUE
C     GET A POINT
      IF(IC .GT. 2) GO TO 210

C     FLAT EARTH
      Y = CONNO*(FLATP - FLATC)
      X = CONEA*(FLONC - FLONP)
      IF(KC .GT.1) GO TO 300
      RETURN
```

```
  210  CONTINUE
       IF(IC .GT. 4) GO TO 220
C      ROTATE COORDINATE SYSTEM SO CENTERED AT PT LO
C ...  IC =  3 --- ROTATE COORDINATE SYSTEM BEFORE PROJECTION
C ...  IC =  4 --- AS 3 BUT ALSO STRETCH TO EQUALIZE ERRORS
       SINLX = SIN(TORAD*FLATP)
       COSLXS = 1. - SINLX*SINLX
       COSLX = SQRT(COSLXS)
       DANG = TORAD*(FLONUS - FLONP)
       SINDA = SIN(DANG)
       COSDA = SQRT(1. - SINDA*SINDA)
       SINLXP = COSLO*SINLX - SINLO*COSLX*COSDA
       XLP = ASIN(SINLXP)
       COSLXP = SQRT(1. - SINLXP*SINLXP)
       SINLAP = SINDA*COSLX/COSLXP
       ALP = ASIN(SINLAP)
       Y = XLP*CLARK
       X = ALP*COSLXP*CLARK
       IF(IC .NE. 4) GO TO  211
C      STRETCH EW COORDINATE TO SPREAD DISTANCE CALCULATION ERROR
C      BETWHEN LATITUDE DISTORTION OF EW DISTANCE AND SHAPE DISTORTION.
       DFLT = ABS(FLATP - FLATUS)*TORAD
       X = X*(1. - DFLT*DFLT/4.)
  211  CONTINUE
       IF(KC .GT.1) GO TO 300
       RETURN
  220  CONTINUE
       IF(IC .GT. 6) GO TO 230


C      FIND COORDINATES FOR A SINGLE POINT FOR ALBERS EQUAL AREA
       SINL = SIN(FLATP*TORAD)
       IF( IC .NE. 6)  GO TO 37
C      SPHERICAL EARTH
       SINBF = SINL
       GO TO 33
   37  CONTINUE
C      ELLIPSOIDAL EARTH
       SINLS = SINL*SINL
       SINBF =SINL*(1. +SINLS*(C1+SINLS*(C2 + SINLS*C3)))/DIV
   33  CONTINUE
       RHOS = RHOHIS + FACT*(SINBHI- SINBF)
       RHO = SQRT(RHOS)
       IF(JC .NE. 1) GO TO 41
       THETA = XN*(CENLON - FLONP)*TORAD
       SINTH = SIN(THETA)
       COSTHS = 1.- SINTH*SINTH
       COSTH = SQRT(COSTHS)
       X    = RHO*SINTH
       Y    = RHOC - RHO*COSTH
       IF(KC .GT.1) GO TO 300
       RETURN
```

```
  230 CONTINUE
      IF (IC .GT. 8) GO TO 240
C     GET A POINT FOR MERCATOR PROJECTION
      COLAT = 1.57079632 - FLATP*TORAO
      IF(IC .EQ.8) GO TO 232
      COSPE = ECC*COS(COLAT)
      AOJ = ((1. + COSPE)/(1. - COSPE))**ECCH
      TANZH = TAN(0.5*COLAT) *AOJ
      GO TO 233
  232 CONTINUE
      TANZH = TAN(0.5*COLAT)
  233 CONTINUE
      FLAM = TORAO*(FLONC - FLONP)
      X = CLARK*FLAM - XO
      Y = CLARK*ALOG(1./TANZH) -YO
      IF( JC .NE. 1) GO TO 131
      IF(KC .GT.1) GO TO 300
      RETURN
  240 CONTINUE
      IF(IC.GT.9) GO TO 250
C     GET A POINT FOR LAMBERT CONFORMAL CONIC PROJECTION
      COLAT = 1.57079632 - FLATP*TORAO
      COSPE = ECC*COS(COLAT)
      ADJ = ((1. + COSPE)/(1. - COSPE))**ECCH
      TANZH = TAN(0.5*COLAT) *AOJ
      RHO = XK*TANZH**XL
      FLAM = TORAD*(CENLN - FLONP)
      SINL = SIN(XL*FLAM)
      COSL =  SQRT(1.-SINL*SINL)
      X = RHO *SINL
      Y = RHOO - RHO*COSL
      IF( JC.NE. 1) GO TO 124
      IF(KC .GT.1) GO TO 300
      RETURN
  250 CONTINUE
      IF (IC .GT. 10) GO TO 270
C     GET A POINT FOR TRANSVERSE MERCATOR PROJECTION
      DIFLN = (CENLO - FLONP)*TORAO
      DIFLNS = DIFLN*DIFLN
      SINLA = SIN(FLATP*TORAO)
      SINLAS = SINLA*SINLA
      COSLAS = 1. - SINLAS
      COSLA = SQRT(COSLAS)
      TANLAS = SINLAS/COSLAS
      ETAS = ECCRTO*COSLAS
      XN = CLARK/SQRT(1. -  ECCSQ*SINLAS)
      PHI = FLATP*TORAO
```

```
      GO TO 291
251   CONTINUE
      X = XN*DIFLN*COSLA*(1. + COSLAS*DIFLNS*((1.- TANLAS + ETAS)/6.
     1 + DIFLNS*COSLAS*(5. - 18.*TANLAS + TANLAS*TANLAS)/120.))
      Y = CLARK*SUM + XN*(DIFLNS*SINLA*COSLA*(0.5 + DIFLNS*COSLAS*
     1 (5. - TANLAS)/24.))
      IF (JC .NE. 1) GO TO 134
      Y = Y - YCNT
      IF(KC .GT.1) GO TO 300
      RETURN
270   CONTINUE
      IF (IC .GT. 11) GO TO 290

C     GET A POINT FOR UNIVERSAL TRANSVERSE MERCATOR PROJECTION
      DIFLN = (GRIDC - FLONP)*3600.
C     REPRESENT EASTERN HEMISPHERE BY NEGATIVE LONGITUDE
      IF(FLONP .LT. 0.) DIFLN = FLONP - GRIDC
      SINLA = SIN(ABS(FLATP)*TORAD)
      SINLAS = SINLA*SINLA
      COSLAS = 1. - SINLAS
      COSLA = SQRT(COSLAS)
      COSLAC = COSLAS*COSLA
      COSLAF = COSLAS*COSLAS
      COSLAT = COSLAS*COSLAC
      TANLAS = SINLAS/COSLAS
      TANLAF = TANLAS*TANLAS
      XNU = CLARKU/SQRT(1.- ECCSU*SINLAS)
      P = 0.0001*ABS(DIFLN)
      PS = P*P
      PC = PS*P
      PF = PS*PS
      PFT = PS*PC
      PSX = PC*PC
      PHI = FLATP*TORAD
      ALPHA = SINLA*COSLA/2.
      SNPR = 1.
      FJ = 1.5*ECCSU
      TJ = 0.5*PHI - ALPHA
      SUM = PHI + FJ*TJ
      DO 272 J = 2,4
      XJ = J
      FJ = FJ*ECCSU*(XJ + 0.5)/XJ
      SNPR = SNPR*SINLAS
      TJ = -SNPR*ALPHA/XJ + (XJ - 0.5)*TJ/XJ
      SUM = SUM + FJ*TJ
272   CONTINUE
      SUM = SUM*(1. - ECCSU)
      FCTR1 = CLARKU*SUM*XKO
      FCTR2 = 0.5E8*XKO*XNU*SINLA*COSLA*SINSS
      FCTR3 = XKO*1.E16*SINSF*SINLA*COSLAC*(5. - TANLAS +  9.*ECCPS*
     1 COSLAS + 4.*ECCPF*COSLAF)*XNU/24.
      FCTR4  =  XNU *SINS*XKO*1.E4*COSLA
      FCTR5  = SINSC*XNU*XKO* 1.E12*COSLAC*(1. - TANLAS + ECCPS*COSLAS)
     1  /6.
```

```
            A6 = PSX*SINSSX*XNU*SINLA*COSLAT*(61. - 58.*TANLAS + TANLAF
           1 + 270.*ECCPS*COSLAS - 330.*ECCPS*SINLAS)*XKO*1.E24/720.
            B5 = PFT*SINSFT*XNU*COSLAT*(5. - 18.*TANLAS + TANLAF +
           1 14.*ECCPS*COSLAS - 58.*ECCPS*SINLAS)*XKO*1.E20/120.
            XN = FCTR1 + PS*FCTR2  + PF*FCTR3 + A6
            XEP = FCTR4*P + FCTR5*PC + B5
            IF(FLATP .LT.0.) XN = 10000000. - XN
            IF(DIFLN .LT.0. ) GO TO 275
            XE = XEP + 500000.
            GO TO 276
      275   CONTINUE
            XE = 500000. - XEP
      276   CONTINUE
            X = XE/RETSM
            Y = XN/RETSM
            RHO = XN
            THETA = XE
            IF(KC .GT.1) GO TO 300
            RETURN
      290   CONTINUE
            IF(IC. GT. 12) GO TO 295
      C     GET A POINT FOR POLYCONIC PROJECTION
            PHI = FLATP*TORAD
            SINLA = SIN(PHI)
            SINLAS = SINLA*SINLA
            COSLA = SQRT(1. - SINLAS)
            ROOT = SQRT(1. - ECCSQ*SINLAS)
            RHO = CLARK*COSLA/(SINLA*ROOT)
            THETA = (CENLO - FLONP)*TORAD*SINLA
      291   CONTINUE
            ALPHA = SINLA*COSLA/2.
            SNPR = 1.
            FJ = 1.5*ECCSQ
            TJ = 0.5*PHI - ALPHA
            SUM = PHI + FJ*TJ
            DO 292 J = 2,4
            XJ = J
            FJ = FJ*ECCSQ*(XJ + 0.5)/XJ
            SNPR = SNPR*SINLAS
            TJ = -SNPR*ALPHA/XJ + (XJ - 0.5)*TJ/XJ
            SUM = SUM + FJ*TJ
      292   CONTINUE
            SUM = SUM*(1. - ECCSQ)
            IF(IC. EQ. 10)GO TO 251
            SS = A1MES *SUM  + RHO
            SINTH = SIN(THETA)
            COSTH = SQRT(1.-SINTH*SINTH)
            X = RHO *SINTH
            Y = SS - RHO*COSTH
            IF(JC .NE. 1) GO TO 171
            Y = Y - YPC
            IF(KC .GT.1) GO TO 300
            RETURN
      295     CONTINUE
```

```
      IF(IC .GT. I3) GO TO  298

C     GET A POINT FOR TRAPAZOIDAL PROJECTION
      DFLT = FLATP - FLATC
      X =(FLONC - FLONP)*(COSLC - DFLT*TORAD*SINLC)
      Y = STMID*DFLT
      IF(KC .GT.I) GO TO 300
      RETURN
  298 CONTINUE

C     GET A POINT FOR STEROGRAPHIC HORIZON PROJECTION
      SINLA = SIN(FLATP*TORAD)
      COSLA = SQRT(1. - SINLA*SINLA)
      FLAM = (CENLO - FLONP)*TORAD
      SINLON = SIN(FLAM)
      COSLON = SQRT(1. - SINLON)
      BOT = 1. + SINLA*SINLO + COSLA*COSLO*COSLON
      X = TCLRK*SINLON*COSLA/BOT
      Y = TCLRK*(SINLA*COSLO - SINLO*COSLA*COSLON)/BOT
      IF (KC .GT. I) GO TO 300
      RETURN

  300 CONTINUE

C     ADJUST OUTPUT
C     START ASSUMING X AND Y ARE IN STATUTE MILES FROM FLATC AND FLONC
      IF (KC .GT. 2) GO TO 310

C     GRID COORDINATES IN STATUTE MILES
C     KC = 2
      X =(X + XGDST)/50.
      Y = ( Y + YGDST)/50.
      RETURN
  310 CONTINUE
      IF(KC .GT. 3) GO TO 320
C     CONVERT TO INCHES FOR MAP PLOTTING
C     KC = 3
      X = X*SMIIN*SCLEW - XMPOF
      Y = Y*SMIIN*SCLNS - YMPOF
      RETURN
  320 CONTINUE
      IF (KC .GT. 4) GO TO  330

C     RECTANGULAR COORDINATES IN NAUTICAL MILES
C     KC = 4
      X = X *SMINM
      Y = Y*SMINM
      RETURN
  330 CONTINUE
      IF( KC.GT.5) GO TO 340
```

```
          C      GRID COORDINATES IN NAUTICAL MILES
          C      KC = 5
                 X =(X + XGDST )*SMINM/60.
                 Y = (Y + YGDST )*SMINM/60.
775              RETURN
          340    CONTINUE
          C      PLOT MAP FOR A SMALL AREA OFFSET FROM THE CENTER OF THE
780       C      COORDINATE SYSTEM.
          C      KC = 6
                 GO TO (341,15),KC6SW
          341    CONTINUE
                 X = (X - XCENT) * SMIIN*SCLEW - XMPOF
785              Y = (Y - YCENT)*SMIIN*SCLNS - YMPOF

                 RETURN
                 END
ENTRY WOULD HAVE RESULTED IN BETTER OPTIMIZATION
```

SUBROUTINE BIMOM (Deck #14)

A General

This subroutine receives tract data described by rectangular coordinates x_i, y_i, and a value P_i, and computes various statistical moments of the ensemble. The original axis are translated to the center of gravity of the value and then rotated to principle axis, i.e., those axis WHICH maximize and minimize the variance along the rotated axis. About these axis all combinations of moments through the fourth are computed. ~~Spherical~~ SPECIAL calculations of moments are made for one tract and two tract cities.

In a second pass through the subroutine the direction of the principle axis as well as moments are known. This enables calculation of several additional quantities. The option allows performing n-sigma rejection of points, calculating the range along the principle axis, calculating the χ^2 statistic~~s~~, and calculating fractional moments.

B Requirements on Calling Program

The communication with the external word through the block common (~~L L STAT~~ ELSTAT). The following description of variables indicates which ~~input~~ INPUT variables are input and which are output for the subroutine. JCTL determines type of subroutine entry.

JCTL = 1 initialize for new city

2 add a point to the sums

3 compute moments

4 add a point in post moment pass

5 finish post moment pass (requires a previous call with JCTL = ~~1~~ 4 (requires a previous call with JCTL = 3 to define α, etc.)

~~KLDN~~ KDEN control of 1 and 2 tract area calculation

~~KCTL~~ KDEN = 0 - for 1 and 2 tract cities set undefined variances to zero

1 - for 1 and 2 tract cities determines variances so city area is filled.

KSIG - Control of n SIGMA rejection

KSIG = 0 - No action

= 1 - Do 5 SIGMA sections of outlying points, i.e., reject points if the distance from either principal area is more than 5 x th corresponding standard deviation

2 - as KSIG = 1 but replace 5 by SGOVTR

This calculation requires a call with JCTL = 1 to initialize followed by calls for each point with JCTL = 4 after a call with JCTL = 3.

KFRAC = 0 - No action

1 - Compute fractional mombents.

This computation requires calls for each point with JCTL = 4 as well as a call with JCTL = 5. If KSIG = 0 no call with JCTL = 1 is needed between the JCTL = 3 call and the JCTL = 4 calls.

KCHI = 0 - No action

1 - Compute χ^2 statistic for calls with JCTL = 4

Compute normal approximation to χ^2 with JCTL = 5.

SGOUTR - value of n is n-SIGMA rejection if JCTL = 4 and KSIG = 2 MUST BE AVAILABLE AT CALL WITH JCTL ≥ 1

XI - Input value of X (East) coordinate for JCTL = 2 or 4

YI - Input value of Y (north) coordinate

PI - Input value (population) associated with point

OUTPUT

IREJT - used in n-SIG rejection JCTL = 4 KSIG ≠ 0

0 last point was not rejected

1 last point was rejected

IREJCT - number of points rejected

CHISQ - X^2 statistic for city comparing with Gaussian distributio

ZAPPX - Z statistic (distance from mean n units of standard deviation) in normal approximation to X^2 statistic

SX...SYYYY - Values of sums of input values. Number of X's in power X is raised to, number of Y's in power Y is ~~rejected~~ raised to, e.g., $SXXXY = \sum_{i=1}^{NTRT} x_i^3 y_i P_i$

Computed in a series of calls with JCTL=2. If these sums are externally determined as well as NTRT and TV, the subroutine can be entered directly with JCTL=3. If a second pass with JCTL = 4 is made, care must be taken that appropriate variables are initialized.

XB center of gravity in original coordinate system in x direction.

YB Center of Gravity in Original Coordinate System in Y direction.

XX Variance in X direction = $\frac{1}{P} \sum_{i=1}^{NTRT} (x_i - \bar{x})^2 P_i$

YY Variance in Y direction

XY Variance in XY direction = $\frac{1}{P} \sum_{i=1}^{NTRT} (x_i - \bar{x}) (y_i - \bar{y}) P_i$

SGXX Standard deviation in x direction

SGYY - Standard deviation in Y direction

SGXY - Standard deviation in XY direction = $XY/\sqrt{SGXX \cdot SGYY}$

TV - Total value = $\sum_{i=1}^{NTRT} P_i$

NTRT - Number of data points

SGRB-SGLLLL- Moments about principle axis. Convention as with SXX... Normalize by dividing by same powers of standard deviations as number of B's or L's.

TWO SEP ~~TWOSLLP~~ - For two tract cities separation between tracts.

TWORAT - For two tract cities ratio of value of tract will be greatest coordinate x (Eastmost) to value of the other tract.

ALPHA - Angle from the NORTH of the principle axis (radians) rotating ~~rotable~~ clockwise. To avoid ambiguity $0 \leq \alpha < \pi$.

ALPHAD - Same as ALPHA but in degress

FRB...FRLLLL - Tractional moments about principle axis. Here the number if B's in L's is the reciprocal of the power, e.g.,

$$FRBBBL = \sum_{i=1}^{NTRT} (x_i{}^3 y_k)^{1/4}/(BGBB^3\ SGLL)^{1/4}$$

where $x' y'$ are coordinates relative to the principle axis.

BMAX - Maximum value of any point along largest principle axis.

BMIN - Minimum value of any point along largest principle axis.

SMAX - Maximum value of any point along smallest principle axis.

SMIN - Minimum value of any point along smallest principle axis.

This subroutine does not require any additional routines except standard FORTRAN ones.

C ALGROITHMS IMPLEMENTED

The algorithm implemented will be derived here briefly. The algebra, while simple, is somewhat NOT tedious. ~~to obtain the final formulae.~~

First, the translation of axis is considered. Let, for example,

$$SXXY \equiv \frac{1}{P} \sum_{i=1}^{NTRT} x_i^2 \, y_i P_i$$

Here, and in similar sums, the power of x_i in the sum is the number of x's in the symbol SXXY, and the power y the number of y's. P is the total value

$$P = \sum_{i=1}^{NTRT} P_i,$$

and NTRT is the number of points considered. For convenience the symbol

$$\sum_{i=1}^{NTRT}$$

will simply be written Σ. We also call $\bar{x}$ = SX and $\bar{y}$ = SY. The moments which are desired are in a coodinate system centered at the center of gravity $\bar{x}$, y, of the original coordinate system and parallel to it. Such momemnts will have the initial letter M. Then

$$MXX = \frac{1}{P} \Sigma \, (x_i - \bar{x})^2 \; P_i = \frac{1}{P} \Sigma x_i{}^2 P_i - \frac{2\bar{x}}{P} \Sigma x_i P_i$$

$$+ \frac{\bar{x}}{P} \Sigma \, P_i = SXX - 2\bar{x}^2 + \bar{x}^2 = SXX - \bar{x}\;^2.$$

In similar fashion

$$MXY = \frac{1}{P} \Sigma \, (x_i - \bar{x}) \, (y_i - \bar{y}) P_i$$

$$= SXY - \bar{x}\,\bar{y}$$

$$MYY = \frac{1}{P} \Sigma \, (y_i - \bar{y})^2 \; P_i$$

$$= SYY - \bar{y}^2$$

$$MXXX = \frac{1}{P} \Sigma \, (x_i - \bar{x})^3$$

$$= SXXX - 3\bar{x} \; SXX + 2 \, \bar{x}^3$$

$$MXXY = \frac{1}{P} \Sigma \; (x_i - \bar{x})^2 \; (y_i - \bar{y}) \; P_i$$

$$= \frac{1}{P} SXXY - 2\bar{x}SXY - \bar{y} \; SXX + 2\bar{x}^2\bar{y}$$

$$MXYY = \frac{1}{P} \Sigma (x_i - \bar{x}) (y_i - \bar{y})^2 P_i$$
$$= SXYY - 3\bar{y}\ SYY + 2\bar{y}^3$$
$$MXXXX = \frac{1}{P} \Sigma (x_i - \bar{x})^4 P_i$$
$$= SXXXX - 4\bar{x}\ SXXX + 6\bar{x}^2\ SXX - 3\bar{x}^4$$
$$MXXXY = \frac{1}{P} \Sigma (x_i - \bar{x})^3 (y_i - \bar{y}) P_i$$
$$= SXXXY - 3\bar{x}\ SXXY + 3\bar{x}^2\ SXY - \bar{y}\ SXXX + 3\bar{x}\ \bar{y}\ SXX - 3\bar{x}^3\bar{y}$$
$$MXXYY = \frac{1}{P} \Sigma (x_i - \bar{x})^2 (y_i - \bar{y})^2 P_i$$
$$= SXXYY - 2\bar{x}\ SXYY - 2\bar{y}\ SXXY + \bar{x}^2\ SYY + \bar{y}^2\ SXX + 4\bar{x}\bar{y}\ SXY - 3\bar{x}^2\bar{y}^2$$
$$MXYYY = \frac{1}{P} \Sigma (x_i - \bar{x}) (y_i - \bar{y})^3 P_i$$
$$= SXYYY - 3\bar{y}\ SXYY + 3\bar{y}^2\ SXY - \bar{x}\ SYYY + 3\bar{x}\bar{y}\ SYY - 3\bar{x}\bar{y}^3$$
$$MYYYY = \frac{1}{P} \Sigma (y_i - \bar{y})^4 P_i$$
$$= SYYYY - 4\bar{y}\ SYYY + 6\bar{y}^2\ SYY - 3\bar{y}^4.$$

In use the subroutine initialized for a particular city by calling it with JCTL = 1. This sets the sums SX, SY, etc., to zero. It is then called for each tract with JCTL = 2 with the values of x_i, y_i, and P_i for that tract. It is then called with JCTL = 3 to compute the ~~first~~ FINAL moments. The first step in this process is to compute the moments with the formulas just given.

Next the principle axis are found. Suppose now x, y are the original axis through the center of gravity, and x',y' are axis rotated counter clockwise through an angle $\bar{\theta}$. Then

$$x' = x \cos \bar{\theta} + y \sin \bar{\theta}$$
$$y' = y \cos \bar{\theta} - x \sin \bar{\theta}$$

Denote moments about the primed axis by a final letter I. Then

$$IXX = \frac{1}{P} \Sigma x_i'^2 P_i = \frac{1}{P} \Sigma (x_i \cos \bar{\theta} + y_i \sin \bar{\theta})^2 P_i$$
$$= \frac{\cos^2\bar{\theta}}{P} \Sigma x_i^2 P_i + 2\frac{\sin \bar{\theta} \cos \bar{\theta}}{P} \Sigma x_i y_i P_i$$
$$+ \frac{\sin^2\bar{\theta}}{P} \Sigma y_i^2 P_i$$
$$= \cos^2\bar{\theta}\ MXX + 2 \sin \bar{\theta} \cos \bar{\theta}\ MXY + \sin^2 \bar{\theta}\ MYY$$

Similarly

$$IXY = \frac{1}{P} \Sigma\, x_i' y_i' P_i$$
$$= (\cos^2 \bar{\theta} - \sin^2\bar{\theta})\, MXY + \sin \bar{\theta}\, (MYY\text{-}MXX)$$

$$IYY = \frac{1}{P} \Sigma\, y_i'^2 P_i$$
$$= \cos^2\bar{\theta}\, MYY - 2 \sin\bar{\theta} \cos\bar{\theta}\, MXY + \sin^2\bar{\theta}\, MXX.$$

It is well known and easily verified that IXX and IYY take on EXTREMAL VALUES ~~external valu~~ when IXY = 0. These give the principle axis we wish. Moreover, if an ellipse is drawn about these principle axis, and the distance, from the origin to the ellipse found at an angle θ to the principle axis, then the moment about the axis is $IxIy/r^2$.

Setting IXY = 0 gives

$$\tan 2\bar{\theta} = -\frac{2MXY}{MYY\text{-}MXX}$$

It shall be assumed $-\frac{\pi}{2} \leq 2\bar{\theta} \leq \frac{\pi}{2}$

It is convenient to use $\theta = - \bar{\theta}$. Then $\sin \theta = - \sin \bar{\theta}$,

$$\cos \theta - \cos \bar{\theta},\ \tan \theta = - \tan \bar{\theta}.$$

Also $x' = x \cos \theta - y \sin \theta$

$$y' = y \cos \theta + x \sin \theta.$$

$$IXX = \cos^2\theta\, MXX\text{-}2 \sin \theta \cos \theta\, MXY + \sin^2 \theta\, MYY$$
$$IYY = \cos^2\theta\, MYY\text{+}2 \sin \theta \cos \theta\, MXY + \sin^2 \theta MXX.$$

We define α as the angle from the Y axis (north) clockwise to the principle axis with its largest value of the moment (semi major axis). Since no direction along the principle axis is specified $0 \leq \alpha < \pi$. Then, it is readily seen that if

$$IXX \geq IYY$$

$$0 \leq \theta \leq \frac{\pi}{4} \quad \alpha = \frac{\pi}{2} + \theta$$

$$-\frac{\pi}{4} \leq \theta \leq 0 \quad \alpha = \frac{\pi}{2} + \theta$$

and if

$$IYY > IXX$$

$$0 \leq \theta \leq \frac{\pi}{4} \quad \alpha = \theta$$

$$-\frac{\pi}{4} \leq \theta \leq 0 \quad \alpha = \pi + \theta$$

In order to determine values for higher moments the formulas for x', y' in terms of x, y and θ are used. It is assumed in these formulas that IXX $\geq$ IYY. The values will be given with B(big) replacing X, and L (little) replacing Y. If IYY <IXX, then after the calculation the roles of B and L are interchanged. We thus have

$$\mathrm{IBB} = \mathrm{IXX}$$

$$\mathrm{ILL} = \mathrm{IYY}$$

$$\mathrm{IBBB} = \frac{1}{P}\,\Sigma\, {x_i'}^3\, P_i = \frac{1}{P}\,\Sigma(x_i \cos\theta - y_i \sin\theta)^3\, P_i.$$

$$= \cos^3\theta\ \mathrm{MXX} - 3\sin\theta\cos^2\theta\ \mathrm{MXXY}$$

$$+\, 3\sin^2\theta\cos\theta\ \mathrm{MXYY} - \sin^3\theta\ \mathrm{MYYY}$$

Similarly

$$\mathrm{IBBL} = \frac{1}{P}\,\Sigma\, {x_i'}^2\, y_i' P_i$$

$$= \cos^3\theta\ \mathrm{MXXY} - 2\sin\theta\cos^2\theta\ \mathrm{MXYY}$$

$$+\, \sin^2\theta\cos\theta\ \mathrm{MYYY} + \sin\theta\cos^2\theta\ \mathrm{MXXX}$$

$$-\, 2\sin^2\theta\cos\theta\ \mathrm{MXXY} + \sin^3\theta\ \mathrm{MXYY}$$

$$\mathrm{IBLL} = \frac{1}{P}\,\Sigma\, x_i' {y_i'}^2\, P_i$$

$$= \cos^3\theta\ \mathrm{MXYY} + 2\sin\theta\cos^2\theta\ \mathrm{MXXY}$$

$$+\, \sin^2\theta\cos\theta\ \mathrm{MXXX} - \sin\theta\cos^2\theta\ \mathrm{MYYY}$$

$$-\, 2\sin^2\theta\cos\theta\ \mathrm{MX}\overset{\mathrm{YY}}{\cancel{\mathrm{XX}}} - \sin\theta\cos^2\theta\ \mathrm{M}\overset{\mathrm{XX}}{\cancel{\mathrm{YY}}}\mathrm{Y}$$

$$\mathrm{ILLL} = \frac{1}{P}\,\Sigma\, {y_i'}^3\, P_i$$

$$= \cos^3\theta\ \mathrm{MYYY} + 3\sin\theta\cos^2\theta\ \mathrm{MXYY}$$

$$+\, 3\sin^2\theta\cos\theta\ \mathrm{MXXY} + \sin^3\theta\ \mathrm{MXXX}$$

$$\mathrm{IBBBB} = \frac{1}{P}\,\Sigma\, {x_i'}^4\, P_i$$

$$= \cos^4\theta\ \mathrm{MXXXX} - 4\sin\theta\cos^3\theta\ \mathrm{MXXXY}$$

$$+\, 6\sin^2\theta\cos^2\theta\ \mathrm{MXXYY} - 4\sin^3\theta\cos\theta\ \mathrm{MXYYY}$$

$$+\, \sin^4\theta\ \mathrm{MYYYY}$$

$$
\begin{aligned}
\text{IBBBL} &= \frac{1}{P} \Sigma \, x_i'^3 \; y_i' P_i \\
&= \cos^4 \theta \text{ MXXXY} - 3 \sin \theta \cos^3 \theta \text{ MXXYY} \\
&+ 3 \sin^2 \theta \cos^2 \theta \text{ MXYYY} - \sin^3 \theta \cos \theta \text{ MYYYY} \\
&+ \sin \theta \cos^3 \theta \text{ MXXXX} - 3 \sin^2 \theta \cos^2 \theta \text{ MXXXY} \\
&+ 3 \sin^3 \theta \cos \theta \text{ MXXYY} - \sin^4 \theta \text{ MXYYY}
\end{aligned}
$$

$$
\begin{aligned}
\text{IBBLL} &= \frac{1}{P} \Sigma \, x_i'^2 \; y_i'^2 \; P_i \\
&= \cos^4 \theta \text{ MXXYY} - 2 \sin \theta \cos^3 \theta \text{ MXYYY} \\
&+ \sin^2 \theta \cos^2 \theta \text{ MYYYY} + 2 \sin \theta \cos^3 \theta \text{ MXXXY} \\
&- 4 \sin^2 \theta \cos^2 \theta \text{ MXXYY} + 2 \sin^3 \theta \cos \theta \text{ MXYYY} \\
&+ \sin^2 \theta \cos^2 \theta \text{ MXXXX} - 2 \sin^3 \theta \cos \theta \text{ MXXXY} \\
&+ \sin^4 \theta \text{ MXXYY}
\end{aligned}
$$

$$
\begin{aligned}
\text{IBLLL} &= \frac{1}{P} \Sigma \, x_i' y_i'^3 \; P_i \\
&= \cos^4 \theta \text{ MXYYY} + 3 \sin \theta \cos^3 \theta \text{ MXXYY} \\
&+ 3 \sin^2 \theta \cos^2 \theta \text{ MXXXY} + \sin^3 \theta \cos \theta \text{ MXXXX} \\
&- \sin \theta \cos^3 \theta \text{ MYYYY} - 3 \sin^2 \theta \cos^2 \theta \text{ MXYYY} \\
&- 3 \sin^3 \theta \cos \theta \text{ MXXYY} - \sin^4 \theta \text{ MXXXY}
\end{aligned}
$$

$$
\begin{aligned}
\text{ILLLL} &= \frac{1}{P} \Sigma \, y_i'^4 \; P_i \\
&= \cos^4 \theta \text{ MYYYY} + 4 \sin \theta \cos^3 \theta \text{ MXYYY} \\
&+ 6 \sin^2 \theta \cos^2 \theta \text{ MXXYY} + 4 \sin^3 \theta \cos \theta \text{ MXXXY} \\
&+ \sin^4 \theta \text{ MXXXX.}
\end{aligned}
$$

Finally, the moments are normalized by dividing by THE appropriate power of the standard deviation. ~~to~~ The power of the larger or small standard deviation in the denominat~~ion~~OR in the same as the number of B's or L's in the symbol. Then, for example

$$\text{SGBBBL} = \text{IBBBL}/(\text{SGBB}^3 \cdot \text{SGLL})$$

where SGBB and SGLL are the standard deviations along the large or small axis.

For one and two moment tracts a special calculation of moments is conducted if KCTL = 2. The product of the two moments if a city is estimated from the following approximate empirical formula

$$\sigma_B \sigma_L = 0.000186 \; P^{0.819}$$

For one tract cities σ_B is set equal to σ_L. Moreover, SGBBBB = SGLLLL = SGBBLL = 3 as for a circular Gaussian distribution. For two tract cities σ_L is computed by dividing $\sigma_B \sigma_L$ by σ_L. Here SGLLLL is set = 3.

For two tract cities a special calculation is carried out to determine the tract separation and ~~rates~~ RATIO of population of the two tracts. Let

$$x_A = \text{distance of Tract A from center of gravity}$$

$$f_A = \text{fraction of total population in Tract A}$$

with analogous definitions for Tract B. Moreover, assume Tract x_A is positive (if not rename the tracts). Then the equation for the moments along the principle axis give

$$1 = f_A + f_B$$

$$0 = x_A f_A + x_B f_B$$

$$\alpha = x_A^2 f_A + x_B^2 f_B$$

$$B = x_A^3 f_A + x_B^3 f_B$$

where

$$\alpha = \text{SGBB}^2$$

$$B = \text{SGBBB} \cdot (\text{SGBB})^3$$

Call

$$R = \frac{B}{2\alpha}$$

and

$$S = \sqrt{\left(\frac{B}{2\alpha}\right)^2 + \alpha}$$

There these equations have the solution

$$x_A = R + S$$

$$x_B = R - S$$

$$f_A = (S - R)/2S$$

$$f_B = (S + R)/2S.$$

The separation TWOSL´P, and ratio of f_A to f_B, TWORAT, are then readily computed.

Several additional calculations may be made by a second pass through the data. This is done by supplying values of x_i, y_i and p_i in succession and calling the subroutine with JCTL = 4.

When the subroutine is called with JCTL = 4 the distances from the principle axis are computed. This is done by setting $\bar{\theta} = \frac{\pi}{2} - \alpha$ and using the rotation equation in $\bar{\theta}$ already presented.

If KSIG ~~KS/G~~ = 1 points are rejected if their distance from the principle axis is greater than SGOUT · SGBB or SGOUT · SGLL. If a point is rejected ith flag IREJCT ~~IRL´JP~~ is set to 1 (it is 0 normally)and the counter IREJCT ~~IRL´JCT~~ is increased by one. The subroutine is then exited and none ~~more~~ of the subsequent calculations are performed for the rejected point. The value of SGOUT is 5 if KSIG = 1 ~~KS/G = 1~~, it is SGOUTR, which must be supplied to the subroutine on the last call with JCTL = 1 if KSIG ~~KS/G~~ = 2. The sums SX . . . SYYYY are incremented for those calls where points are not rejected. If these sums are to be used the subroutine must have been reinitiated by a call with JCTL = 1 just before the calls with JCTL = 4. New principle axis and moments can be computed by a call with JCTL = 3. If desired the entire rejection process can then be repeated.

The maximum and minimum values along the two principle axis BMAX, BMIN, SMAX, and SMIN are computed ~~(note: Greek letter lower case chi)~~.

If KCHI = 1 a χ^2 statistic ,CHISQ is computed. To do this the expected value is computed as

$$e_i = \frac{1}{2\pi \cdot SGBB \cdot SGLL} \; e^{-\frac{x_i'^2}{2VARBB}} \; e^{-\frac{y_i'^2}{2 \cdot VARLL}}$$

The x^2 statistic is then computed as the sum of

$$\frac{(e_i - P_i)^2}{e_i}$$

~~Where~~ Here the number of degrees of freedom is interpreted as the total population. If a call with JCTL = 5 is made after the JCTL =4 calls, a normal approximation to the x^2 statistic is computed. Here

ZAPPX = (CHV - 1)/SGNM

with

CHV = CHISQ/TV

and

SGNM = 2/TV.

If KFRAC = 1 a set of fractional moments are computed. These are computed from sums taken the same as with the normal moments except instead of powers of x' or y', roots of those variables are taken, i.e., the values of x' any y' are raised to the reciprocal powers. Thus, for example

$$FRBBBL = \Sigma(x')^{1/3} y',$$

when the same convention or number of B's and L's is adopted as before. For even power roots the root of the absolute value of the distance is used. In computing FRB, FRL, and FRBL, absolute values are used due to the special nature of the sums.

At the completion of the calls with JCTL = 4, a call with JCTL = 5 will normalize the fractional moments by dividing by the roots of the variances to the same degree as the number of B's or L's in the symbol. Thus, for example

$$FRBBBL = FRBBBL/(SIBBB^{1/3} \cdot SGLL).$$

The interpretation of the fractional moments should be analogous to that of the power moments. They will emphasize those parts of the distribution near the axis, rather than outlying points ~~and~~ as with higher moments.

```
      SUBROUTINE BINOM
C     NEVUNS STANDARD
C     LAST REVISED SEPT. 5, 1973

C     THIS SUBROUTINE TAKES MOMENTS OF POPULATION DATA AND FINDS
C     PRONCIPLE AXIS AND ALL COMBINATIONS OF MOMENTS THROUGH FOURTH
C     DEGREE ABOUT THESE AXIS. INPUT IS RELATIVE TO AN ARBITRARY SET O
C     RECTANGULAR AXIS.  THE PARAMETER JCTL
C     DETERMINES THE TYPE OF ACTION. JCTL = 1 INITIALIZE FOR NEW SET O
C     CALCULATIONS, JCTL = 2 ADD TO SUMS SX, SY ETC. WITH VALUES FROM
C     A SINGLE POINT. JCTL = 3 FIND PRINCIPLE AXIS AND MOMENTS.
C     JCTL = 4 ADD POINTS IN A SECOND PASS, JCTL = 5 CLEANUP SECOND PA
C     A 0 VALUE TURNS OFF THE FOLLOWING SWITCHES
C     IF KDEN = 1 RADIUS OF 1 AND 2 TRACTS CITIES SHOULD BE DETERMINED
C     BY FORMULA.
C     KSIG = 1 DO N-SIGMA REJECTION OF OUTLYING POINTS WITH N =5
C     KSIG = 2 N = SGOUTR
C     KFRAC = 1 COMPUTE FRACTIONAL MOMENTS IN 4 TH AND 5TH PASS.
C     KCHI = 1 COMPUTE CHISQ STATISTIC AND NORMAL APPROX.
C     VALUES OF XB OR YB GREATER THAN 5000 CAUSES 6400 SIGNIFICANT
C     DIGIT CAPACITY TO BE EXCEEDED.


      COMMON/ELSTAT/JCTL,KDEN,KSIG,KFRAC,KCHI,SGOUTR,XI,YI,PI,
     1 IREJT,IREJCT, CHISQ,ZAPPX, SX,SY,SXX,SXY,SYY,SXXX,SXXY,
     2SXYY,SYYY,SXXXX,SXXXY,SXXYY,SXYYY,SYYYY,XB,YB,XX,YY,XY,SGXX,SGX
     3 SGYY, TV, NTRT, SGBB,SGLL,SGBBB,SGBBL,SGBLL,SGLLL,  SGBBBB,
     4 SGBBBL,SGBBLL,SGBLLL,SGLLLL, TWOSEP,TWORAT ,ALPHA, ALPHAD,
     5FRB,FRL,FRBB,FRBL,FRLL,FRBBB,FRBBL,FRBLL,FRLLL,FRBBBB,FRBBBL,
     6FRBBLL,FRBLLL,FRLLLL, BMAX,BMIN,SMAX,SMIN

      DATA PIW,PIH,TWOPI,TODEG,OTH/3.14159265,1.57079632,6.2831853,
     1 57.2957796,0.33333333/

      IF( JCTL .NE. 1) GO TO 20

C     INITIALIZE CALCULATIONS
      TV = 0.
      SX = 0.
      SY = 0.
      SXX = 0.
      SXY = 0.
      SYY = 0.
      SXXX = 0.
      SXXY = 0.
      SXYY = 0.
      SYYY = 0.
      SXXXX = 0.
      SXXXY = 0.
      SXXYY = 0.
      SXYYY = 0.
```

```
      SYYYY = 0.
      NTRT = 0
      CHISQ = 0.
      IREJCT = 0
      BMIN = 99999.
      SMIN = 99999.
      BMAX = 0.
      SMAX = 0.
      FRB = 0.
      FRL = 0.
      FRBB = 0.
      FRBL = 0.
      FRLL = 0.
      FRBBB = 0.
      FRBBL = 0.
      FRBLL = 0.
      FRLLL = 0.
      FRBBBB = 0.
      FRBBBL = 0.
      FRBBLL = 0.
      FRBLLL = 0.
      FRLLLL = 0.
      IF(KSIG .EQ. 2) GO TO 11
      SGOUT = 5.
      GO TO 12
   11 CONTINUE
      SGOUT = SGOUTR
   12 CONTINUE
      RETURN

   20 CONTINUE
      IF (JCTL. NE.2) GO TO  50
   21 CONTINUE

C     ADD A POINT
      NTRT = NTRT + 1
      TV = TV + PI
      XS = XI*XI
      XC = XS*XI
      YS = YI*YI
      YC = YI*YS
      SX = SX + XI*PI
      SY = SY + YI*PI
      SXX = SXX + XS*PI
      SXY = SXY + XI*YI*PI
      SYY = SYY + YS*PI
      SXXX = SXXX + XC*PI
      SXXY = SXXY + XS*YI*PI
      SXYY = SXYY + XI*YS*PI
      SYYY = SYYY + YC*PI
      SXXXX = SXXXX + XS*XS*PI
      SXXXY = SXXXY + XC*YI*PI
      SXXYY = SXXYY + XS*YS*PI
      SXYYY = SXYYY + XI*YC*PI
```

```
      SYYYY = SYYYY + YS*YS*PI
      IF( JCTL .EQ. 4) GO TO 81
      RETURN
 50   CONTINUE
      IF( JCTL .NE. 3) GO TO 80

C     NOW FIND MOMENTS

      IF(NTRT  .GT. 1) GO TO 45
      IF(NTRT  .EQ. 1) GO TO 46
      RETURN
 46   CONTINUE

C     ONE TRACT ONLY GETS 0. VALUES FOR MOMENTS
C     IF OTHER VALUES ARE DESIRED THEY MAY BE INSERTED EXTERNALLY
      XB = SX/TV
      YB = SY/TV
      XX      =0.
      XY      =0.
      YY      =0.
      IF(KDEN .EQ.1) GO TO 47
      SGXX    =0.
      SGXY    =0.
      SGYY    =0.
      SGBB    =0.
      SGLL    =0.
      SGBBBB  =0.
      SGBBLL  =0.
      SGLLLL  =0.
      GO TO 48
 47   CONTINUE
      SGPROS = 0.000186*TV**0.819
      SGPRO = SQRT(SGPROS)
      SGXX = SGPRO
      SGYY = SGPRO
      SGBB = SGPRO
      SGLL = SGPRO
      SGBBBB = 3.
      SGBBLL = 3.
      SGLLLL = 3.
 48   CONTINUE
      SGBBB   =0.
      SGBBL   =0.
      SGBLL   =0.
      SGLLL   =0.
      SGBBBL  =0.
      SGBLLL  =0.
      ALPHA = 0.
      ALPHAD=0.
      SINN = 0.
      COSN = 1.
      RETURN
 45   CONTINUE
```

```
C     FIRST NORMALIZE SUMS BY TOTAL POPULATION
      XB = SX/TV
      YB = SY/TV
      XBS = XB*XB
      XBC = XBS*XB
      YBS = YB*YB
      YBC = YBS*YB
      SXX = SXX/TV
      SXY = SXY/TV
      SYY = SYY/TV
      SXXX = SXXX/TV
      SXXY = SXXY/TV
      SXYY = SXYY/TV
      SYYY = SYYY/TV
      SXXXX = SXXXX/TV
      SXXXY = SXXXY/TV
      SXXYY = SXXYY/TV
      SXYYY = SXYYY/TV
      SYYYY = SYYYY/TV
C     NOW TRANSLATE AXIS
      XX = SXX - XBS
      YY = SYY - YBS
      XY = SXY - XB*YB
      XXX = SXXX - 3.*XB*SXX + 2.*XBC
      YYY  = SYYY - 3.*YB*SYY + 2.*YBC
      XXY = SXXY - 2.*XB*SXY - YB*SXX + 2.*XBS*YB
      XYY = SXYY - 2.*YB*SXY - XB*SYY + 2.*XB*YBS
      XXXX = SXXXX - 4.*XB*SXXX + 6.*XBS*SXX -3.*XBS*XBS
      YYYY = SYYYY - 4.*YB*SYYY + 6.*YBS*SYY - 3.*YBS*YBS
      XXXY = SXXXY - 3.*XB*SXXY + 3.*XBS*SXY - YB*SXXX + 3.*XB*YB*SXX
     1 - 3.*XBC*YB
      XYYY =  SXYYY - 3.*YB*SXYY + 3.*YBS*SXY - XB*SYYY + 3.*XB*YB*SYY
     1 - 3.*XB*YBC
      XXYY = SXXYY - 2.*XB*SXYY - 2.*YB*SXXY + XBS*SYY + YBS*SXX
     1 + 4.*XB*YB*SXY - 3.*XBS*YBS
      XXYY = SXXYY - 2.*XB*SXYY - 2.*YB*SXXY + XBS*SYY + YBS*SXX
     1 + 4.*XB*YB*SXY - 3.*XBS*YBS
      SGXX = SQRT(ABS( XX))
      SGYY = SQRT(ABS( YY))
      SGXY = SQRT(ABS( XY))
      IF(  XY .LT. 0.) SGXY = - SGXY

C     NOW FIND PRINCIPAL AXIS
      IF (ABS(YY - XX) . GT . 0.00001) GO TO 52
      TTHETA = PIH
      GO TO 53
   52 CONTINUE
        TTHETA = ATAN(2.*XY/(YY - XX))
   53 CONTINUE
      THETA = TTHETA/2.
      SINO = SIN(THETA)
```

```
      COSO = COS(THETA)
      SINS = SINO*SINO
      SINC = SINO*SINS
      SINF = SINS*SINS
      COSS = COSO*COSO
      COSC = COSS*COSO
      COSF = COSS*COSS
      VARXX = XX*COSS - 2.*SINO*COSO*XY + YY*SINS
      VARYY = YY*COSS + 2.*SINO*COSO *XY + XX*SINS

C     FIND ELLIPSE ANGLE ALPHA
      IF( VARXX .LT. VARYY) GO TO 55
      VARBB = VARXX
      VARLL = VARYY
      ALPHA = PIH + THETA
      GO TO 58
 55   CONTINUE
      VARLL = VARXX
      VARBB = VARYY
      IF( THETA .LT. 0.) GO TO 56
      ALPHA = THETA
      GO TO 58
 56   CONTINUE
      ALPHA = PIH+ THETA
 58   CONTINUE
      ALPHAD = ALPHA*TODEG
      THN = PIH - ALPHA
      SINN = SIN(THN)
      COSN = COS(THN)
      SGBB = SQRT(ABS(VARBB))
      SGLL = SQRT(ABS(VARLL))
      IF( VARLL .LT. 0.) SGLL = 0.

C     NOW ROTATE AXIS FOR HIGHER MOMENTS
C     ASSUME FOR THE MOMENT THAT ROTATED X AXIS IS IUG AXIS
      SGBBB = XXX*COSC -3.*XXY*COSS*SINO + 3.*XYY*COSO*SINS
     1 - YYY*SINC
      SGLLL = YYY*COSC +3.*XYY*COSS*SINO + 3.*XXY*COSO*SINS+ XXX*SIN
      SGBBL = XXY*COSC - 2.*XYY*SINO*COSS + YYY*SINS*COSO +XXX*SINO*
     1 COSS -2.*XXY*SINS*COSO + XYY*SINC
      SGBLL = XYY*COSC + 2.*XXY*SINO*COSS + XXX*SINS*COSO - YYY*SINO*
     1 COSS - 2.*XYY*SINS*COSO - XXY*SINC
      SGBBBB = XXXX*COSF - 4.*XXXY*COSC*SINO + 6.*XXYY*COSS*SINS
     1 - 4.*XYYY*COSO*SINC + YYYY*SINF
      SGLLLL = YYYY*COSF + 4.*XYYY*COSC*SINO + 6.*XXYY*COSS*SINS
     1 + 4.*XXXY*COSO*SINC + XXXX*SINF
      SGBBBL = XXXY*COSF - 3.*XXYY*COSC*SINO + 3.*XYYY*COSS*SINS
     1 - YYYY*COSO*SINC + XXXX*SINO*COSC - 3.*XXXY*COSS*SINS
     2 + 3. *XXYY*COSO*SINC - XYYY*SINF
      SGBLLL = XYYY*COSF + 3.*XXYY*COSC*SINO + 3.*XXXY*COSS*SINS
     1 + XXXX*COSO*SINC - YYYY*COSC*SINO - 3.*XYYY*COSS*SINS
     2 - 3.*XXYY*COSO*SINC - XXXY*SINF
      SGBBLL = XXYY*COSF - 2.*XYYY*SINO*COSC + YYYY *SINS*COSS
```

```
     1  + 2.*XXXY*SINO*COSC - 4.*XXYY*SINS*COSS + 2.*XYYY*SINC*COSO
     2  + XXXX*SINS*COSS - 2.*XXXY*SINC*COSO + XXYY*SINF
        IF ( VARXX .GE. VARYY) GO TO 61
C       INTERCHANGE AXIS SINCE Y AXIS IS LARGER NOT X
        TEMP = SGBBB
        SGBBB = SGLLL
        SGLLL = TEMP
        TEMP = SGBBL
        SGBBL = SGBLL
        SGBLL = TEMP
        TEMP = SGBBBB
        SGBBBB = SGLLLL
        SGLLLL = TEMP
        TEMP = SGBLLL
        SGBLLL = SGBBBL
        SGBBBL = TEMP
 61     CONTINUE

C       NORMALIZE MOMENTS
        IF( SGBB .LT. 0.0000001) GO TO 46
        IF(  SGLL .LT. 0.0000001) GO TO 65
        SGBBB = SGBBB/(VARBB*SGBB)
        SGBBL = SGBBL/(VARBB*SGLL)
        SGBLL = SGBLL/(SGBB*VARLL)
        SGLLL = SGLLL/(VARLL*SGLL)
        SGBBBB = SGBBBB/(VARBB*VARBB)
        SGBBBL = SGBBBL/(VARBB*SGBB*SGLL)
        SGBBLL = SGBBLL/(VARBB*VARLL)
        SGBLLL = SGBLLL/(SGBB*VARLL*SGLL)
        SGLLLL = SGLLLL/(VARLL*VARLL)
        GO TO 66
 65     CONTINUE
        SGBBB = SGBBB/(VARBB*SGBB)
        SGBBBB = SGBBBB/(VARBB*VARBB)
        SGBBL   =0.
        SGBLL   =0.
        SGLLL   =0.
        SGBBBL  =0.
        SGBBLL  =0.
        SGBLLL  =0.
        SGLLLL  =0.
 66     CONTINUE
        IF (NTRT  .NE. 2) GO TO 68
        IF( VARBB .LT. 0.0000001) GO TO 68

C       SPECIAL CALCULATION FOR TWO TRACTS
        BETA = SGBBB*VARBB*SGBB
        RATO = BETA/(2.*VARBB)
        ROOT = SQRT(RATO*RATO + VARBB)
C       THIS ASSUMES THAT XA IS LARGER THAN XB ALONE THE POSITIVE B AXIS
        FSA = (ROOT - RATO)/(2.*ROOT)
        FSB = (RATO + ROOT)/(2.*ROOT)
```

```
      C     TWORAT IS THE RATIO OF POPULATION OF THE MOST POSITIVE / THE OTH
            TWORAT = FSA/FSB
            TWOSEP = 2.*ROOT
      67    CONTINUE
            IF(KDEN .NE. 1) GO TO 68
            SGPROS = 0.000186*TV**0.819
            SGLL = SGPROS/SGBB
            SGLLLL = 3.
            SGBBL=0.
            SGBLL=0.
            SGLLL=0.
            SGBBBL=0.
            SGBBLL=0.
            SGBLLL=0.
      68    CONTINUE
            RETURN
      80    CONTINUE
            IF( JCTL .GT. 4) GO TO 90

      C     SECOND PASS COLLECTION OF DATA
      C     GET COORDINATES RELATIVE TO PRINCIPLE AXIS
            XU = XI - XB
            YU = YI - YB
            XP = XU*COSN + YU*SINN
            YP = YU*COSN- XU*SINN
            XPA = ABS(XP)
            YPA = ABS(YP)
            IF(XPA .LT. 0.0000001) GO TO 86
            XPSN = XP/XPA
            GO TO 87
      86    CONTINUE
            XPSN = 1.
      87    CONTINUE
            IF(YPA .LT. 0.0000001) GO TO 88
            YPSN = YP/YPA
            GO TO 89
      88    CONTINUE
            YPSN = 1.
      89    CONTINUE
            XPS = XP*XP
            YPS = YP*YP
            XPSR = SQRT(XPA)
            YPSR = SQRT(YPA)
            XPCR = XPSN*(XPA)**OTH
            YPCR = YPSN*(YPA)**OTH
            IF( KSIG . EQ. 0) GO TO 81

      C     THREE SIGMA REJECTION TEST
            IREJT = 0
            BLMT = SGOUT*SGBB
            SLMT = SGOUT*SGLL
            IF( XPA.LE.BLMT .AND.YPA.LE. SLMT) GO TO 21
```

```
          IREJT = 1
          IREJCT = IREJCT + 1
          RETURN
   81     CONTINUE
          IF( XP .GT. SMAX) SMAX = XP
          IF (YP .GT. SMAX) SMAX = YP
          IF (XP .LT. SMIN) SMIN = XP
          IF( YP .LT.SMIN) SMIN = YP
          IF(KCHI .NE. 1) GO TO 83
C         CHI SQUARE STATISTIC CALCULATION
          IF(VARBB .LT. 0.0000001) GO TO 96
          TMPA = EXP(-XP*XP/(2.*VARBB))
          GO TO 97
   96     CONTINUE
          TMPA = 1.
   97     CONTINUE
          IF(VARLL .LT. 0.0000001) GO TO 98
          TMPB = EXP(-YP*YP/(2.*VARLL))
          GO TO 99
   98     CONTINUE
          TMPB = 1.
   99     CONTINUE
          VALEX = TV*TMPA*TMPB/TWOPI
          TEMP = VALEX-PI
          CHISQ = CHISQ + TEMP*TEMP/VALEX
   83     CONTINUE
          IF (KFRAC .NE. 1) GO TO 85
C         COMPUTE SUMS FOR FRACTIONAL MOMENTS
          FRB = FRB + XPA*PI
          FRL = FRL + YPA*PI
          FRBB = FRBB + XPSR*PI
          FRBL = FRBL + XPA*YPA*PI
          FRLL = FRLL + YPSR*PI
          FRBBB = FRBBB + XPCR*PI
          FRBBL = FRBBL + XPSR*YP*PI
          FRBLL = FRBLL + XP*YPSR*PI
          FRLLL = FRLLL + YPCR*PI
          FRBBBB = FRBBBB + PI*XPA**0.25
          FRBBBL = FRBBBL + XPCR*YP*PI
          FRBBLL = FRBBLL + XPSR*YPSR*PI
          FRBLLL = FRBLLL + XP*YPCR*PI
          FRLLLL = FRLLLL + PI*YPA**0.25
   85     CONTINUE
          RETURN

   90     CONTINUE

C         FINAL CLEANUP CALCULATIONS FROM SECOND PASS
          IF( KCHI .NE. 1) GO TO 91
C         COMPUTE NORMAL APPROXIMATION TO CHI SQUARE DISTRIBUTION.
          CHV = CHISQ/TV
```

```
      SGNM = 2./TV
      ZAPPX = (CHV - 1.)/SGNM
   91 CONTINUE
      IF(KFRAC .NE. 1) GO TO 92
C     NORMALIZE FRACTIONAL MOMENTS
      IF(NTRT .EQ. 1) GO TO 93
      IF(SGBB .LT. 0.0000001) GO TO 93
      SBBSR = SQRT(SGBB)
      SBBCR = (SGBB)**OTH
      IF(NTRT .EQ.2) GO TO 94
      IF (SGLL .LT. 0.0000001) GO TO 94
      SLLSR = SQRT(SGLL)
      SLLCR = (SGLL)**OTH
      FRB = FRB/(SGBB*TV)
      FRL = FRL/(SGLL*TV)
      FRBB = FRBB/(SBBSR*TV)
      FRBL = FRBL/(SGBB*SGLL*TV)
      FRLL = FRLL/(SLLSR*TV)
      FRBBB = FRBBB/(SBBCR*TV)
      FRBBL = FRBBL/(SBBSR*SGLL*TV)
      FRBLL = FRBLL/(SGBB*SLLSR*TV)
      FRLLL = FRLLL/(SLLCR*TV)
      FRBBBB = FRBBBB/(TV*SGBB**0.25)
      FRBBBL = FRBBBL/(SBBCR*SGLL*TV)
      FRBBLL = FRBBLL/(SBBSR*SLLSR*TV)
      FRBLLL = FRBLLL/(SGBB*SLLCR*TV)
      FRLLLL = FRLLLL/(TV*SGLL**0.25)
      GO TO 92
   93 CONTINUE
      FRB = 0.
      FRL = 0.
      FRBB = 0.
      FRBL = 0.
      FRLL = 0.
      FRBBB = 0.
      FRBBL = 0.
      FRBLL = 0.
      FRLLL = 0.
      FRBBBB = 0.
      FRBBBL = 0.
      FRBBLL = 0.
      FRBLLL = 0.
      FRLLLL = 0.
      GO TO 92
   94 CONTINUE
      FRBB = FRBB/(SBBSR*TV)
      FRBBB = FRBBB/(SBBCR*TV)
      FRBBBB = FRBBBB/(TV*SGBB**0.25)
      FRL = 0.
      FRBL = 0.
      FRLL = 0.
      FRBBL = 0.
      FRBLL = 0.
```

```
          FALLL = 0.
          FRBBAL = 0.
          FRBALL = 0.
          FRBLLL = 0.
500       FALLLL = 0.
       92 CONTINUE
          RETURN
          END
MORY WOULD HAVE RESULTED IN BETTER OPTIMIZATION
```

SUBROUTINE TELFAR (Deck #15)

A. GENERAL

This subroutine determines the minimum distance between two ellipses. It was developed to determine if two population nodes, represented by elliptical normal distributions, are close enough to be considered contiguous. The location, size, and orientation of the two ellipses are given to the subroutine. If the minimum distance is less than the input parameter DCUT, the value of DCUT is returned in the output parameter DSTMN. Otherwise, DSTMN contains the minimum distance between the ellipses. By setting DCUT equal to zero the minimum distance between the ellipses is always returned. However, the calculation time may be appreciably longer than when DCUT is larger than this minimum distance. If the ellipses intersect and DCUT = 0, DSTMN will be a value near zero but not necessarily identically zero.

B. REQUIREMENTS ON CALLING PROGRAM

The following variables in the common block/ELPAR/ must be supplied by the calling program.

FLATA, FLONA, FLATB, FLONB - the latitude and longitude of the centers of the two ellipses denoted by A and B.

SIGBA, SIGLA, SIGBB, SIGLB - half the semimajor and semiminor axis of the two ellipses. The size of the actual ellipse for which the distances are calculated has semimajor and semiminor axis twice the length input. Distances are in statute miles.

ALPHA, ALPHB - the angle in degrees clockwise from the north to the ellipse semimajor axis. These angles are between 0 and 180

degrees. DCUT - the minimum separation distance in statute miles of interest. If the ellipses are closer than DCUT, a value of DCUT is returned.

The subroutine returns the distance in statute miles in the variable DSTMN. If a value of -999 is returned no convergence was obtained in 100 iterations.

C. ALGORITHMS IMPLEMENTED

The minimum distance is determined by searching through values along the semimajor axis if the two ellipses, x_1 and x_2, until successive distance calculations are less than an error tolerance TOL. The ellipse furthest west is numbered ellipse 1. An $\bar{x}$, $\bar{y}$ coordinate system is centered in ellipse 1 with $\bar{x}$ pointing north and $\bar{y}$ west. A point x, y, on ellipse 1 is expressed in $\bar{x}$ $\bar{y}$ coordinates by rotation through an angle α. For ellipse two, coordinates are rotated through an angle α and then translated. The square of the distance is found as the sum of the squares of the difference in $\bar{x}$ and $\bar{y}$ coordinates.

The points on the ellipse closest to the other ellipse is used. Call θ the clockwise angle from the north from the center of ellipse 1 to the center of ellipse 2. Then if $\alpha_1 < \theta$ the y_1 coordinate is negative, otherwise positive. If $\alpha_2 < \theta$ the y_2 coordinate is positive, otherwise it is negative. From a value of x_i the value of y_i is readily calculated by $/y_i/ = b_i \sqrt{1-x_i^2/a_i^2}$, where a_i and b_i are the semimajor and semiminor axis.

The initial values of x_1 and x_2 are taken as 0. Values of distances squared are computed for $x_i = x_i - \Delta x_i$, x_i, and $x_i + \Delta x_i$, where $\Delta x_i = .01a_i$, holding the x coordinate on the other ellipse

constant. The first and second derivatives of the distances squared as a function of x are estimated. A parabolic fit is made to estimate the value of x_1 for which the distances squared are minimized. A new value of x_1 is taken where the change in x_1 is the constant "speed" multiplied by the old value. The process is then repeated until the change in value of distance squared is less than the value TOL.

In the current subroutine SPEED = 0.5 and TOL = 0.01. Typical solutions require from 10 to 20 iterations.

```
      SUBROUTINE TELFAR
C     NEVUNS STANDARD
C     LAST REVISIVED AUGUST 15,1973.

C     A SUBROUTINE TO FIND THE DISTANCE BETWEEN TWO ELLIPSES.

C     THE DISTANCE IS FOUND BY BRUTE FORCE MINIMIZATION USING THE TWO
C     X DISTANCES ALONG THE ELLIPSE AS INDEPENDENT VARIABLES.
C     FLATA, ETC ARE CG, STD DEV ALONG LARGE AND SMALL AXIS AND
C     ANGLE ROTATED FROM NORTH FOR ELLIPSE A, FLATB ETC. FOR ELLIPSE B.
C     DSTMN IS MINIMUM DISTANCE  BETWEEN ELLIPSES
C     DCUT IS MINIMUMSEPERATION OF INTEREST. IF DISTANCE IS LESS THAN
C     DCUT ROUTINE IS EXITED WITH DSTMN EQUAL TO DCUT.

      COMMON/ELPAR/FLATA,FLONA, SIGBA,SIGLA,ALPHA,FLATB,FLONB,SIGBB,
     1SIGLB,ALPHB,DCUT,DSTMN
      DATA CONV,TORAD,PIH/69.1713,0.017453,1.570796325/
C     INITIALIZE
      TOL = 0.01
      SPEED = 0.5
      KNT = 0
      DCUTSQ = DCUT*DCUT
C     DENOTE ELLISPE FURTHEST TO THE WEST BY SUBSCRIPT O,OTHER BY T
      IF(FLONB .GT. FLONA) GO TO 10
      FLATO = FLATA
      FLONO = FLONA
      AO = 2.*SIGBA
      BO = 2.*SIGLA
      ALPHO =   ALPHA*TORAD
      FLATT = FLATB
      FLONT = FLONB
      AT = 2.*SIGBB
      BT = 2.*SIGLB
      ALPHT =    ALPHB*TORAD
      GO TO 11
   10 CONTINUE
      FLATO = FLATB
      FLONO = FLONB
      AO = 2.*SIGBB
      BO = 2.*SIGLB
      ALPHO =   ALPHB*TORAD
      FLATT = FLATA
      FLONT = FLONB
      AT = 2.*SIGBA
      BT = 2.*SIGLA
      ALPHT =   ALPHA*TORAD
   11 CONTINUE
      AOQ = 0.25*AO
      ATQ = 0.25*AT
```


PRECEDING PAGE BLANK-NOT FILMED

```
              AOSQ = AO*AO
              ATSQ = AT*AT
              AOM = -AO
              ATM = -AT
              AOQM = -AOQ
              ATQM = -ATQ
              DELXC = (FLATT - FLATO)*CONV
              AVGL = 0.5*TORAD*(FLATT + FLATO)
              DELYC = (FLONO-FLONT)*CONV*COS(AVGL)
              DELYCA = ABS(DELYC)
              IF(DELYCA .LT. 0.00001)DELYC=0.00001
              TANTH = DELXC/DELYC
              THETA = PIH - ATAN(TANTH)
              IF(ALPHO .GT.THETA) GO TO 15
              SIDO = -1.
              GO TO 16
        15    CONTINUE
              SIDO = 1.
        16    CONTINUE
              IF(ALPHT .GT.THETA) GO TO 17
              SIDT = 1.
              GO TO 18
        17    CONTINUE
              SIDT = -1.
        18    CONTINUE
              COSAO = COS(ALPHO)
              SINAO = SIN(ALPHO)
              COSAT = COS(ALPHT)
              SINAT = SIN(ALPHT)
              XOOLD = 0.
              XTOLD = 0.
              DELXO = .01*AO
              DELXT = .01*AT
              DOLD = 9999999.
              XO = XOOLD
              XT = XTOLD
              IGO = 1
              GO TO 100
        21    CONTINUE
              DCNT = DSQ

        20    CONTINUE
        C     ITERATE ON XOOLD AND XTOLD
        C     FIND SLOPES AND CURVATURE OF DIST CHANGE IN TWO DIRECTIONS
              XO = XOOLD + DELXO
              XT = XTOLD
              IGO = 2
              GO TO 100
        22    CONTINUE
              DPLSQ = DSQ
              XO = XOOLD - DELXO
              IGO = 3
              GO TO 100
        23    CONTINUE
```

```
      DMNSO = DSQ
      XO = XOOLD
      XT = XTOLD + DELXT
      IGO = 4
      GO TO 100
   24 CONTINUE
      DPLST = DSQ
      XT = XTOLD - DELXT
      IGO = 5
      GO TO 100
   25 CONTINUE
      DMNST = DSQ

      DDO = DPLSO + DMNSO - 2.*DCNT
      DELO = .5*(DPLSO-DMNSO)
      DDT = DPLST + DMNST - 2.*DCNT
      DELT = .5*(DPLST - DMNST)
      IF(DDO .LT. 0.0001)DDO = 0.0001
      IF(DDT .LT. 0.0001)DDT = 0.0001
      DISTO = -SPEED*DELO*DELXO/DDO
      DISTT = -SPEED*DELT*DELXT/DDT
      IF(DISTO .GT.AOQ)DISTO = AOQ
      IF(DISTT .GT . ATQ)DISTT=ATQ
      IF(DISTO .LT.AOQM) DISTO = AOQM
      IF(DISTT .LT.ATQM) DISTT = ATQM
      XOTRY = XOOLD + DISTO
      IF(XOTRY .GE. AO) GO TO 31
      IF(XOTRY .LE. AOM) GO TO 32
      XOOLD = XOTRY
      GO TO 33
   31 CONTINUE
      XOOLD = 0.5*(XOOLD + AO)
      GO TO 33
   32 CONTINUE
      XOOLD = 0.5*(XOOLD + AOM)
   33 CONTINUE
      XTTRY = XTOLD + DISTT
      IF(XTTRY .GE . AT) GO TO 34
      IF(XTTRY .LE.ATM) GO TO 35
      XTOLD = XTTRY
      GO TO 36
   34 CONTINUE
      XTOLD = 0.5*(XTOLD + AT)
      GO TO 36
   35 CONTINUE
      XTOLD = 0.5*(XTOLD + ATM)
   36 CONTINUE
      XO = XOOLD
      XT = XTOLD
      IGO = 6
      GO TO 100
   26 CONTINUE
      DCNT = DSQ

      IF(DCNT .GT. DCUTSQ) GO TO 41
```

```
            DSTMN = DCUT
      C     DISTANCE SMALL ENOUGH SO EXIT
            RETURN
      41    CONTINUE
            DIFF = DOLD - DCNT
            DOLD = DCNT
            KNT = KNT + 1
            IF(KNT .LT.100) GO TO 38
            DSTMN = - 999.
            RETURN
      38    CONTINUE
            IF(ABS(DIFF) .GT. TOL) GO TO 20
            DSTMN = SQRT(DCNT)
            RETURN

      100   CONTINUE
      C     ENTER WITH X1,X2 AND GET DISTANCE
            IF(XO .GE. AO) XO = AO - 0.0001
            IF(XO .LE.AOM) XO = AOM + 0.0001
            IF(XT .GE.AT) XT = AT - 0.0001
            IF(XT .LE. ATM) XT = ATM + 0.0001
            YO = SIDO*BO*SQRT(1. - XO*XO/AOSQ)
            YT = SIOT*BT*SQRT(1. - XT*XT/ATSQ)
      C     XBAR AXIS POINTS NORTH,YBAR AXIS POINTS WEST.
            XOB = XO*COSAO + YO*SINAO
            YOB = -XO*SINAO + YO*COSAO
            XTB = XT*COSAT + YT*SINAT
            YTB = -XT*SINAT + YT*COSAT
            DX = XTB + DELXC -XOB
            DY = YOB + DELYC - YTB
            DSQ = DX*DX + DY*DY
            GO TO (21,22,23,24,25,26 ),IGO
            END
```

Subroutines FALLYB (DECK #16)
FALLWB (DECK #17)
FALLDB (DECK #18)
FALLCB (DECK #19)

A. GENERAL

These subroutines are an implementation of the WSEG 10/NAS Modified Fallout Model. The equations implemented are contained in "An Analysis of the Fallout Prediction Models presented at the USNRDL-DASA Fallout Symposium of September 1962, Volume 1: Analysis,Comparison, and Classification of Models," M. Polon, Ford Instrument Company, USNRDL-TRC-68, September 8, 1966, Unclassified.

The calculations directly complement the equations given in the reference. A copy of the reference is included with the equations implemented referenced by a vertical bar. The subroutines compute quantities sensitive to ~~various~~ INPUT parameters INSERT A and place them in a storage array FOR LATER USE. The definition of these parameters follows. The subroutine FINALLY will ~~also~~ calculate WSEG BIOLOGICAL DOSES, Biological Doses at ~~this~~ T HOURS, and maximum Biological Doses as given in the National Military Command System Support Center SIDAC Model described in "Single Integrated Damage Analysis Capability (SIDAC)(U), Analytical Manual, Ralph D. Mason, National Military Command System Support Center, CSMAM67-68, October 18, 1968, CONFIDENTIAL." These subroutines are normally called in the sequence given.If calling parameters for a previous subroutine are unchanged, this subroutine need not be recalled for a new dose calculation if the storage array parameters are externally supplied.

INSERT B -- NEXT PAGE

The subroutine FALLDB compute~~d~~s downwind distance dependent parameters. If the control parameter MDCAL is 0 only WSEG Biological Dose is calculated. If the control parameter is 1, then the NMCSSC time dependent doses are calculated; if 2, also the maximum dose is calculated. In the latter two cases the time of weapon detonation in hours after the start of the war must be supplied.

A of yield, fission fraction and height of burst, wind velocity and wind shear; downwind distance, and crosswind distance when called

~~The subroutine FALLWB computes wind dependent and wind shear dependent parameters~~.

The subroutine FALLCB computes crosswind dependent parameters and return FINAL answers. The answer locations are the following elements of the storage array.

Element Number	Information
31	Time if NMCSSC max. dose
32	H + 1 dose rate
33	Equivalent WSEG Biological Dose
34	NMCSSC dose at 7 hrs.
35	NMCSSC dose at 22 hrs.
36	NMCSSC dose at 68 hrs.
37	NMCSSC dose at 211 hrs.
38	NMCSSC dose at 800 hrs.
39	Maximum NMCSSC biological dose.

INSERT B

The subroutine FALLYB computes parameters dependent on yield, fission fraction, and height of burst. If external calculation of effects of fission fraction or height of burst are described, these parameters may be set equal to 1 and 0, respectively. A 0 height of burst will bypass the height of burst calculation. This calculation produces an adjustment factor given by

$$AF = 0.5(1-x)^2(2+x) + .001x ,$$

where $x = HOB/(180 \cdot (1{,}000 \cdot YIELD)^{0.4})$.

The subroutine FALLWB computes wind dependent and wind shear dependent parameters.

B. Requirements on Calling Program

The calling program is required to manage the common block /FLWSEG/. The input parameters are AS listed BELOW. The subroutine output fills some of the elements in the output array. For example the subroutine FALLYB uses as input values of YLDFW (Weapon Yield), FISSFW (Weapon Fission Fraction), and HOBEW (High Burst). It provides values of ARRYFW(1) to ARRYFW(6) which are used, but never modified by the SUBSEQUENT subroutines. Thus, if several weapons of the same yield are used, the subroutine FALLYB need only be called once for this group of weapons provided the calling program insures the appropriate values are in elements 1 to 6 of the array ARRYFW.

WORD NO. IN BLOCK	NAME	MEANING	SUBROUTINE REQUIRED
1	YLDFW	Weapon Yield (MT)	FALLYB
2	FISSFW	Weapon Fission Fraction	FALLYB
3	HOBFW	Height of Brust (ft)	FALLYB
4	WNDFW	Fallout Wind Speed (mph)	FALLWB
5	SHRFW	Wind Shear (mph/ft)	FALLWB
6	DWDFW	Downwind Distance (statute mi)	FALLDB
7	MDCAL	Control Parameter Value 0 - WSEG Bio Dose only 1 - NMCSSC Time Dose also 2 - NMCSSS Max Dose also	FALLDB & FALLCB
8	TWPNFW	Time of Weapon Detonation, only needed if MDCAL $\neq$ 0	FALLDB
9	CWDFW	Cross Wind Distance (statute mi)	FALLCB
10	ARRYFW(40)	Storage Array	ALL

DEFINITIONS OF QUANTITIES IN ARRAY FOR USE IN THE WSEG 10/NAS IMPLEMENTATION

Input: Y - yield HOB - Height of Burst F - Fission FRACTION ~~Effect~~
W - wind D - Downwind distance C - Crosswind distance
S_c - shear T_w - Time of weapon detonation

AY MENT	NAME	SYMBOL	MEANING	FORMULA
1	SIGO	σ_σ	Effective cloud diameter	$\exp[0.70+\ln Y/3-3.25/(4\cancel{Y}+(\ln Y+5.4)^2)]$
2	SIGOS	σ_o^2	Square of cloud diameter	σ_o^2
3	HC	H_c	Height of cloud center	$44+6.1\ \ln Y-.205(\ln Y+2.42)$ $\lvert \ln\ 2.42 \rvert$
4	SIGH	σ_H	Vertical thickness	$0.18\ H_c$
5	TCHAR	T	Characteristic arrival time	$T=1.0573203(\frac{12}{60}H_c-2.5(\frac{H_c}{60})^2)\ \cdot$ $(1-0.5\ \exp((-H_c/25)^2))$
6	NFAC	F	Normalization factor	2,000,000 FISS·AF·Y where AF=0 if $x(\equiv HOB/(180\cdot(1000Y)^{0.4}) > 180$ $AF=0.5(1-x)^2(2+x)+\ .001x \quad 0<x\leq 180$ $AF=1.0 \quad x=0$
7	SIGCFA	σ_{ca}	σ_c factor	$8/L\ \sigma_o^2$ where $L_o = WT$ $\sigma_u^2 = \sigma_o^2(8L_o^2+\sigma_o^2)/(2L_o^2+\sigma_o)$ $L^2 = L_o^2 + 2\sigma_u^2$
8	SIGCFB	σ_{cb}	σ_c factor	$(L_o + T\cdot\sigma_H\cdot S_c/L^2)^2$
9	SIGCFC	σ_{cc}	σ_c factor	$\sigma_o^2+2(\sigma_u\cdot T\cdot\sigma_H\cdot S_c/L)^2$
0	XL	M		$(L_o^2 + \sigma_u^2)/(L_o^2 + 0.5\ \sigma_u^2)$
1	GFAC	g_1	Part of function g	$1./(L\cdot\Gamma(1+\frac{1}{M}))$ where $\Gamma=$ gamma function $M>1.002$ $\Gamma=\ 1 \quad M\leq 1.002$
2	TAFA	T_{oA}	T_o factor	$L_o^2T^2/(L^2(L_o^2+0.5\ \sigma_u^2)$

RAY EM NT	NAME	SYMBOL	MEANING	FORMULA
13	TAFB	T_{aB}	T_a factor	$0.25 + 2\sigma_u^2/(L_o^2 + 0.5\,\sigma_u^2)$
14	SIGU	σ_u	UPWIND spread factor	$\sigma_o^2(L_o^2 + 8\sigma_o^2)/(L_o^2 + 2\sigma_o^2)$
15	SIGFD	8/L	Multiplication factor	8/L
16	~~A2FA~~ ALFA	2/W	Multiplication factor	2/W
17	~~A2FB~~ ALFB		Multiplication factor	$.001 \cdot H_c \cdot W/\sigma_o$
18	FDF		FD factor	$L_o/(L\,\alpha_1\sigma_u)$ where $\alpha_1 = 1/(1.+.001\,H_cW/\sigma_o)$
19	Not Used			
20	FCFB		FC factor	$1/(\sqrt{2\pi}\,\sigma_c$ where σ_c^2 = SIGCFA·DF + (SIGCFB·DF) + SIGCFC and DF = min ~~$(8/D + 2\sigma_u/L, 1, 3)$~~ $(8/D + 2\sigma_u 1/L, 3)$
21	FCFB		FC factor	$0.5/(\alpha_2 \cdot \sigma_c)^2$ where $\alpha_2 = 1/(1.+\text{ALFB}(1-\text{cumnor}(2D/W)))$
22	FD	F_d	Downwind intensity	NFAC·GT·cumnor(FDF·D) where GT = GFAC·exp$(-D/L)^n$
23	TA	T_a	Time of fallout arrival	$(\text{TAFA}(D+2\sigma_u)^2 + \text{TAFB})^{1/2}$
24	B10	B	Ratio of WSEG Biological Dose to H+1 Dose Rate	$\exp-(.287 + .52\ln(T_a/31.6) + .04475(\ln T_a/31.6)^2$
25 26 27 28 29	DBT~~M~~I	D_{BTi}	NMCSSC Biological Dose factor	$(T_a - (T_i - T_w)^{-0.2})^Z$ where $Z = .5+4.5\exp(-.00061+.00025\,T_a^{-.2}(T_i - T_a)$ and $T_1 = 7$, $T_2=22$, $T_3=68$, $T_4=211$, $T_5=2800$ hrs.
30	DBTM	D_{BTm}	Max-Value of Dose Factor	Parabolic interpolation from DBT_i
31	TM	T_m	Time of max dose	Parabolic interpolation from DBT_i
32	DR	D_{H+1}	~~[illegible]~~ H+1/dose rate	$F_c \cdot F_d$ where $F_c = \text{FCFB} \cdot \exp(-\text{FCFB} \cdot C^2)$

ARRAY ELEMENT	NAME	SYMBOL	MEANING	FORMULA
33	ERD	D_{B10}	WSEG Biological Dose	$DR \cdot B10$
34, 35, 36, 37, 38	DRTI	D_{RT_i}	NMCSSC Dose Rate at T_i hours	$DBT_i \cdot D_{H+1}$
39	DRM	D_{RM}	Max NMCSSC Biological Dose	$D_{DTm} \cdot D_{H+1}$

Elements	Calculated from
1-6	Yield
7-18	Wind, Shear and Elements 1-~~13~~ 6
20-31	Downwind Distance, Time of Weapon Detonation and Elements 1-18
32-39	Crosswind Distance and Elements 1-31

USNRDL-TRC-68
8 September 1966

AN ANALYSIS OF THE FALLOUT PREDICTION MODELS
Presented at the USNRDL–DASA Fallout Symposium of September 1962
Volume I: Analysis, Comparison, and Classification of Models

by

M. Polan

Ford Instrument Company
Division of Sperry Rand Corporation
31–10 Thomson Avenue
Long Island City, New York 11101

Contract N228(62479)62185

U.S. NAVAL RADIOLOGICAL DEFENSE LABORATORY

SAN FRANCISCO · CALIFORNIA · 94135

5.2 WEAPON SYSTEMS EVALUATION GROUP, DOD (WSEG)* (Reference 6)

5.2.1 Cloud Submodel

1. Early Dynamics. The WSEG model is used for land- and water-surface bursts. Early dynamics are not explicitly represented, but the current model contains ad hoc corrections (see α_1 and α_2 under Transport, subsection 5.2.2) to reduce the area covered by fallout prior to the stabilization of the cloud. The model is used for yields in the range between 1 KT and 100 MT.

2. Cloud Geometry. The cloud has diffuse dimensions that thin out according to a Gaussian distribution which is given below, under Cloud Activity Distribution. The effective cloud diameter (D), height of center (H_c), and vertical thickness (ΔH) used in the program are given below.

$$D = 4\,\sigma_o$$

where σ_o (st. mi.) $= \exp\left[0.70 + (\log_e Y)/3 - 3.25/\left(4.0 + (\log_e Y + 5.4)^2\right)\right]$

or σ_o (kft) $= \exp\left[0.061 + (\log_e W)/3 - 3.25/\left(4.0 + (\log_e W - 1.51)^2\right)\right]$

where Y = total yield in MT
W = total yield in KT

$$H_c\ (\text{kft}) = 44 + 6.1 \log_e Y - 0.205\,(\log_e Y + 2.42)\,\left|\log_e Y + 2.42\right|$$

*The version of the WSEG model described in this section (5.2) is the version embodied in the 1962 FORTRAN subroutine listed in Vol. 1 of the Symposium Compilation. This version is currently being used by WSEG and other agencies (see Section 5.3); it differs significantly from the version in WSEG RM-10 (Ref. 6), which is sometimes incorrectly referenced as the basis of specific fallout predictions.

That the WSEG model is perhaps the least understood model in current use is due to (1) the way in which WSEG RM-10 was written and (2) the lack of documentation for model revisions subsequent to publication of WSEG RM-10. Much of the explanatory material in this section is based on discussions between Dr. Pugh and the author during the April 1966 OCD-DASA Fallout Phenomena Symposium. It is hoped that the material presented herein will provide a clear, although condensed, statement of the model.

or $H_c(kft) = 1.86 + 6.1 \log_e W - 0.205 (\log_e W - 4.49) |\log_e W - 4.49|$

$\Delta H(kft) = 4\ \sigma_H$

where $\sigma_H = 0.18\ H_c$.

The heights of the effective cloud top (1.36 H) and bottom (0.64 H_c) as functions of yield are shown in Figures 6.2 and 6.3. The effective cloud diameter as a function of yield is shown in Figure 6.4.

3. Cloud Activity Distribution. The activity is postulated be distributed within the cloud according to the density function ρ:

$$\rho(DWD, CWD, H) = \frac{1}{(2\pi)^{3/2}\sigma_o^2 \sigma_H} \exp -\left[\frac{DWD^2 + CWD^2}{\sigma_o^2} + \left(\frac{H - H_c}{\sigma_H}\right)^2\right]$$

The symbols used in this equation are defined under Transport in subsection 5.2.2. The resulting vertical distributions of activity for a 20-KT and a 20-MT burst are shown in Figure 6.6. This density function is not used explicitly in the program, but it is the basis of the function developed to compute the dose rate at a point, which is discussed under Transport.

4. Activity—Particle-Size Distribution. This characteristic was never included explicitly in the model. The first RAND distribution, m = 4.49, σ = 0.69, was used to develop the rate of cloud deposition function, g(t), used in the version of the model described in WSEG RM-10 (Ref. 6). The 1962 version uses an entirely different g(t) function, which was derived without consideration of any such size distribution (see Particle Setting Rates, subsection 5.2.2).

5. Normalization. The value of the normalization factor NF used is 2400 r/hr per KT/sq. st. mile. The outputs are computed directly from this value without a reduction for terrain shielding.

The quantity 2400 represents fission and induced activity on fallout particles, and excludes activity in the gaseous or monomolecular state. A factor F, the fraction of the fallout included within the close-in pattern is discussed, but the absence of this factor from the FORTRAN program effectively assigns it a value of 1. This was done to reflect the preponderance of fallout activity associated with particles greater than 20 microns in diameter.

5.2.2 Transport Submodel

1. Wafers. The cloud is not divided into finite elements.

2. Particle Settling Rates. Particle settling rates were never explicitly included in the model. Instead, the following g(t)

function representing the fraction of cloud activity deposited on the ground per unit time as a function of time was developed:

$$g(t) = \frac{\bar{F}}{T\Gamma(1 + \frac{1}{n_o})} \exp\,(-t/T)^{n_o}$$

where $\bar{F}$ is the close-in fallout fraction previously specified under "Normalization," t is time after burst in hours, Γ is the complete gamma function which serves to maintain normalization for different values of n_o, T is the characteristic time for the rate of deposition to fall to 1/e of its initial value, and n_o is a dimensionless parameter controlling the deposition rate for late fallout. Note that the above g(t) function is not the g(t) function that actually appears in the RM-10 or in the current version of the model; the latter g(t) is presented subsequently in this discussion.

Empirical expressions for T and n_o that bring the above g(t) function into correspondence with theoretical settling rates for the RAND particle sizes are given in the original RM-10. The supplement to RM-10 contains a modified expression for T which is a fit to data on the rate of fallout deposition from one 1951 and three 1956 nuclear tests.

The expression for T used in the current model is:

$$T \text{ (hours)} = 1.0573203 \left[\frac{12}{60} H_c - 2.5\left(\frac{H_c}{60}\right)^2\right] \left[1 - 0.5 \exp -\left(\frac{H_c}{25}\right)^2\right]$$

where H_c is in kilofeet.

The RM-10 expression for n_o, also in terms of H_c, allows n_o to vary from about 1 to 1.5. A National Academy of Science committee subsequently deemed the varying n_o to be an unnecessary refinement in view of the large scatter and uncertainty in the data. Consequently, n_o was set equal to 1, and does not now appear in the model. A consequence of these changes in T and n_o is that the derivation of the current WSEG model is entirely independent of assumptions regarding particle sizes or settling rates.

Before g(t) is used in the RM-10 or in the current version of the model, it is changed from a "time" to a "distance" function, as follows: During the derivation of g(t), the fraction of cloud activity landing is expressed per unit time, as a function of time after burst. In the g(t) function actually in the model, the fraction landing is expressed per unit distance, as a function of distance from GZ along the hotline (the hotline in this case extending both upwind and downwind of GZ). The following is the g(t) function in the current model:

$$g(t) = \frac{1}{L\,\Gamma(1+\frac{1}{n})} \exp\left(-\frac{|DWD|}{L}\right)^n$$

where DWD, n, and L are as defined in detail under Transport, below.

It may be noted here that since L is approximately the downwind distance (DWD) given by EFW · T, and t = DWD/EFW, then DWD/L ≈ DWD/EFW · T ≈ t/T. The use of L instead of T in the denominator of the expression for g(t) provides the "per unit distance" instead of the "per unit time" in this function. The dimensionless parameter n is intended to provide a smooth transition to a symmetrical upwind versus downwind pattern as EFW → 0. In the current version of the model, n ≈ 1, except when EFW → 0.

3. Winds. This model uses a one-vector effective fallout wind (EFW) in mph that represents the mean horizontal wind between H_c and the ground, and a crosswind shear (S_c) in mph per kft of cloud thickness. The latter represents the change in the wind components normal to the EFW over the vertical extent of the cloud (from $H_c - 2 \cdot \sigma_H$ to $H_c + 2\sigma_H$) divided by the vertical thickness of the cloud ($4\,\sigma_H$). The EFW and the S_c are constant in time and space for each burst.

A downwind shear (S_d) is used in the RM-10 version of the model. It was subsequently set equal to zero; that is, S_d is omitted in the current model, since for typical values of EFW, it produced negligible effects, and as EFW → 0, it erroneously increased the areas within the exposure rate contours.

4. Transport. The ideal plane exposure rate normalized to H+1 hours (DR) can be computed for any point (DWD, CWD) on the ground as the product of a downwind transport function (f_d) and a crosswind transport function (f_c).

$$\underline{DR}\ \text{(roentgens/hour)} = f_d \cdot f_c$$

where

$$f_d = W \cdot NF \cdot F \cdot \bar{F} \cdot g(t)\ \phi\left(\frac{L_o \cdot DWD}{L \cdot \alpha_1\,\sigma_d}\right)^* \quad \text{and} \quad f_c = \frac{1}{\sqrt{2\pi}\,\sigma_c} \exp -\frac{1}{2}\left(\frac{CWD}{\alpha_2\,\sigma_c}\right)^2$$

*Although $\bar{F}$ is not in the program for the current version of the model, it is included in this expression because WSEG for the past five to six years has applied a factor of 5/6 to the exposure rate and the accumulated exposure outputs of this program. In effect, this changes $\bar{F}$ from 1 to 5/6. (Pugh, G.E., personal communication, 26 April 1966.)

and

W = the total yield in KT

NF = the normalization factor, $NF = 2400$ r/hr per KT/sq. st. mile

F = the fission-to-total-yield ratio

$g(t)$ = the fraction of cloud activity landing per unit distance as a function of distance from GZ along the hotline (discussed under Particle Settling Rates)

ϕ = the standard cumulative normal function. It serves to provide some upwind fallout from the pre-stabilized cloud and to avoid a discontinuity at DWD = 0.

L_o = the distance (in st. mi.) a particle moves downwind with the effective fallout wind EFW during the characteristic time T

$$L_o = EFW \cdot T$$

L = a modified form of L_o. The use of L instead of L_o in some parts of the downwind transport equation permits a more consistent mathematical treatment of the effect of the downwind extent of the stabilized cloud.

$$L = (L_o^2 + 2\,\sigma_u^2)^{1/2}$$

DWD = downwind distance (in st. mi.), positive in the direction of EFW, negative in the opposite direction

CWD = the crosswind distance (in st. mi.), positive in the direction 90° clockwise from EFW, negative in the direction 90° counterclockwise from EFW

α_1 = an empirical adjustment in f_d to reduce the area covered by fallout prior to cloud stabilization, reflecting (1) the small size of the cloud at early times and (2) the tendency for the toroidal circulation in the cloud to sweep particles inward.

kft?

$$\alpha_1 = \frac{1}{1 + \dfrac{0.001\,H_c \cdot EFW}{\sigma_o}}$$

α_2 = the equivalent of α_1 in the crosswind transport function. The factor $(1 - \phi\,(2\,DWD/EFW))$ serves to phase out α_2 at about the end of 2 hours.

$$\alpha_2 = \frac{1}{1 + \frac{0.001 \cdot H_c \cdot EFW}{\sigma_o}\left(1 - \phi\left(\frac{2 \cdot DWD}{EFW}\right)\right)}$$

n = a dimensionless parameter in the g(t) "distance" function that provides a transition to a symmetrical upwind versus downwind pattern as EFW⟶0. In the current version:

$$n = \frac{L_o^2 + \sigma_u^2}{L_o^2 + 0.5 \cdot \sigma_u^2}$$

σ_u and σ_c = parameters relating to the upwind and crosswind spread of the pattern, respectively

$$\sigma_u^2 = \sigma_o^2 \frac{(L_o^2 + 8\sigma_o^2)}{L_o^2 + 2\sigma_o^2}$$

$$\sigma_c^2 = \sigma_o^2 + \left(\frac{8\,|DWD + 2\sigma_u|\,\sigma_o^2}{L}\right) + 2\left(\frac{\sigma_u \cdot T \cdot \sigma_H \cdot S_c}{L}\right)^2$$

$$+ \left(\frac{(DWD + 2\sigma_u)\, L_o \cdot T \cdot \sigma_H \cdot S_c}{L}\right)^2, \text{where } \frac{8\,|DWD + 2 \cdot \sigma_u|}{L} \text{ is always} \leq 3$$

5.2.3 Output Submodel

1. Summing. The exposure rate at and the accumulated exposure by any time after an average time of arrival of fallout are based on the $t^{-1.2}$ rule. The average time of arrival of fall is computed from the following expression.

$$T_a(\text{hours}) = \left(0.25 + \frac{L_o^2\,(DWD + 2\sigma_u)^2\,T^2}{L^2\,(L_o^2 + 0.5 \cdot \sigma_u^2)} + \frac{2 \cdot \sigma_u^2}{L_o^2 + 0.5\sigma_u^2}\right)^{1/2}$$

The following approximation is used to compute the maximum effective biological exposure. It is based on 10% of the exposure causing irreparable damage, and a 30-day time-constant for the reparable portion.

$$ERD = BIO\ \underline{DR}$$

where

$$BIO = \exp - \left[0.287 + 0.52 \log_e\left(\frac{T_a}{31.6}\right) + 0.04475 \log_e\left(\frac{T_a}{31.6}\right)^2\right]$$

The above expression for BIO is a higher-order fit to the same data to which $(19/T_a)^{1/3}$, the RM-10 equivalent, is a lower-order fit.

2. Contours. Contours can be constructed as follows. An exposure rate (or exposure) value and a set of DWD's are selected for input to the program. The program will then compute the corresponding CWD's.

```
      SUBROUTINE FALLYB
C     NEVUNS STANDARD.
C     LAST REVISION ON OCT. 27, 1972.
C     TO COMPUTE THE YIELD DEPENDENT PARAMETERS IN THE WSEG 1/-NAS
C     FALLOUT MODEL.   YIELD IS YIELD IN MEGATONS, FISS IS FISSION
C     FRACTION, HOB IS HEIGHT OF BURST IN FEET, ARRY IS AN ARRY OF FORTY
C     ELEMENTS USED TO PRESERVE RESULTS OF DIFFERENT SUBROUTINE CALLS.
C     SUBROUTINES SHOULD BE CALLED IN ORDER OF NEW YIELD, WIND VELOCITY
C     OR WIND SHEAR, DOWNWIND DISTANCE, AND CROSSWIND DISTANCE.
C     THE VALUES IN ARRY(1) TO ARRY(6) ARE FILLED HERE.
      COMMON/FLWSER/YIELD,FISS,HOB,EFW,SC,DWD,MOCAL,TWPN,CWD,ARRY(40)
      XLNY = ALOG(YIELD)
      TEM=XLNY+5.4
      TEMP=0.70+0.3333333* XLNY-3.25/(4.0+TEM*TEM)
      ARRY (1) = EXP(TEMP)
      ARRY(2) = ARRY(1) *ARRY(1)
      TEMP = XLNY +2.42
      ARRY (3) = 44.+ 6.1*XLNY -0.205*TEMP*ABS(TEMP)
      ARRY(4) = 0.18*ARRY(3)
       HCTWO = ARRY(3)/25.
      HCSIX = ARRY(3)/60.
      ARRY(5)=1.0573203*(12.*HCSIX-2.5*HCSIX*HCSIX)*(1.0-0.5*EXP(-HCTWO*
     1HCTWO))
C     HEIGHT OF BURST SENSITIVE CALCULATIONS
C     HOB = 0. BYPASSES HOB CALCULATION AND ALLOWS OUTSIDE CONTROL.
      IF(HOB .GT. 0.) GO TO 5
      ARRY(6) = 2000000.*FISS*YIELD
      RETURN
    5 CONTINUE
      XMHB=180.*(YIELD*1000.)**0.4
      IF(HOB.LE.XMHB) GO TO 10
      ARRY(6)=0.
      RETURN
   10 CONTINUE
      TEMP=HOB/XMHB
      AF=0.5*(1.-TEMP)*(1.-TEMP)*(2.+TEMP)+0.001*TEMP
C     FISS = 1 ALLOWS OUTSIDE CONTROL OF FISSION FRACTION
      ARRY(6)=2000000.*FISS*AF*YIELD
      RETURN
      END
```


PRECEDING PAGE BLANK-NOT FILMED

```
      SUBROUTINE FALLWB
C     NEVUNS STANDARD.
C     LAST REVISION ON OCT. 27, 1972.
C     TO COMPUTE THE WIND SPEED OR WIND SHEAR EFFECTS IS THE WSEG 10-
C     NAS FALLOUT MODEL.
C     EFW IS WIND VELOCITY IN STATUTE MILES PER HOUR, SC IS WIND SHEAR
C     IN STATUTE MILES PER HOUR PER KILOFOOT, ARRY IS A STORAGE ARRAY.
C     THE VALUES IN ARRY(7) TO ARRY(18) ARE SUPPLIED HERE.
      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWD,MDCAL,TWPN,CWD,ARRY(40)
       XLO = EFW*ARRY(5)
      XLOS = XLO*XLO
      SIGUS=ARRY(2)*(XLOS+8.*ARRY(2))/(XLOS+2.*ARRY(2))
      ARRY(14) = SQRT(SIGUS)
      XLS = XLOS + 2.*SIGUS
      XL = SQRT(XLS)
       ARRY(15) = 8./XL
      ARRY(7)=ARRY(15)*ARRY(2)
      TMPA = ARRY(5)*ARRY(4)*SC
      TEMP = XLO*TMPA/XLS
      ARRY(8) = TEMP*TEMP
      TEMP = ARRY(14) *ARRY(5)*ARRY(4)*SC/XL
      ARRY(9) = ARRY(2) +2.*TEMP*TEMP
      XLOPS = XLOS + 0.5*SIGUS
       ARRY(10) = (XLOS + SIGUS)/XLOPS
      IF(ARRY(10).LT.1.002) GO TO 8
      TM = 1./ARRY(10)
C     GAMMA FUNCTION  APPROX HASTINGS P.156
      GAMMA=1. +TM*(-0.57669867 + TM*(0.97781781+TM*(-0.8235627+TM*(
     10.67399080 + TM*(-0.3282793 + TM*0.07673206)))))
      ARRY(11)=1./(XL*GAMMA)
      GO TO 9
    8 ARRY(11)=1./XL
    9 CONTINUE
      ARRY(17) = 0.001*ARRY(3)*EFW/ARRY(1)
      ALONE = 1./(1. + ARRY(17))
      ARRY(18) = XLO/ (XL*ALONE*ARRY(14))
       ARRY(12) = XLOS *ARRY(5) *ARRY(5)/(XLS*XLOPS)
      ARRY(13) = 0.25 + 2.*SIGUS/XLOPS
      IF(EFW.LT.0.000001) GO TO 5
      ARRY(16) = 2./EFW
      GO TO 6
    5 ARRY(16)=999999999.
    6 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FALLDB
C     NEVUNS STANDARD.
C     LAST REVISION ON OCT. 27, 1972.
C     TO COMPUTE DOWNWIND DISTANCE EFFECTS IN THE WSEG 10 - NAS FALLOUT
C     MODEL.
C     DWD IS THE DOWNWIND DISTANCE IN STATUTE MILES, MOCAL = 0-ONLY
C     WSEG BIO DOSE, 1-NMCSSC BIO DOSE AT BHRS, 2-ALSO MAX DOSE.
C     TWPN IS TIME OF WEAPON DETONATION IN HOURS, ARRY IS A STORAGE
C     ARRAY.
C     THE VALUES IN ARRY(20) TO ARRY(31) ARE COMPUTED HERE.
      DIMENSION BHRS(5)
      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWD,MOCAL,TWPN,CWD,ARRY(40)
      DATA BHRS(1),BHRS(2), BHRS(3), BHRS(4), BHRS(5)/
     1 7.,22., 68., 211., 800./
      TP=DWD+2.*ARRY(14)
      DWP=ABS(TP)
      TMP = ARRY(15) *DWP
      IF(TMP.LE.3.) GO TO 8
      DWP=3./ARRY(15)
    8 CONTINUE
      SIGCS = ARRY(9) + ARRY(7) *DWP + ARRY(8)*TP*TP
      SIGC = SQRT(SIGCS)
      TA = ARRY(16)*DWD
      IF(TA.GT.4.) GO TO 11
C     APPROX HASTINGS P.185 FOR CUM NOR
      TM = ABS(TA/1.414213562)
      TMP = 1. + TM*(0.278393+TM*(0.230389+TM*(0.000972+TM*0.078108)))
      TMP = TMP *TMP
      CUP = 1.-1./(TMP*TMP)
      IF(TA .LT. 0.)GO TO 6
      CUV = 0.5*(1. + CUP)
      GO TO 7
    6 CONTINUE
      CUV = 0.5*(1. - CUP)
    7 CONTINUE
      ALTWO = 1./(1.+ARRY(17)*(1.-CUV))
      GO TO 12
   11 ALTWO=1.
   12 CONTINUE
      ARRY(20)=1./(2.50663*SIGC)
       TMP = ALTWO *SIGC
      ARRY(21) = 0.5/(TMP*TMP)
      TA = DWD *ARRY(18)
      IF(TA.LT.5.) GO TO 14
      CUV=1.
      GO TO 15
   14 CONTINUE
C     APPROX HASTINGS P.187 FOR CUM NOR.
      TM = ABS(TA/1.414213562)
      TMP =1.+TM*(0.0705230784+TM*(0.0422820123+TM*(0.0092705272+TM*
```

```
         1(0.0001520143+TM*(0.0002765672+TM*0.0000430638)))))
          TMP=TMP*TMP
          TMP=TMP*TMP
          TMP=TMP*TMP
          TMP=TMP*TMP
          CUP= 1.-1./TMP
          IF(TA .LT. 0.)GO TO 21
          CUV = 0.5*(1. + CUP)
          GO TO 22
   21     CONTINUE
          CUV = 0.5*(1. - CUP)
   22     CONTINUE
       15 CONTINUE
          IF(ARRY(10).LT.1.002) GO TO 17
           TMP =(ABS(OWO)*ARRY(15)/8.)**ARRY(10)
          GO TO 18
       17 TMP=ABS(OWO)*ARRY(15)/8.
       18 CONTINUE
          IF(TMP.LT.30.) GO TO 19
          ARRY(22) = 0.
          ARRY(23) = 999999.
          ARRY(23) = 999999.
          RETURN
   19     CONTINUE
          GT =ARRY(11)*EXP(-TMP)
          ARRY(22) = ARRY(6)*GT*CUV
          TMP = ARRY(13) + TP * TP*ARRY(12)
          ARRY(23) = SQRT(TMP)
          TMP = ALOG(ARRY(23)/31.6)
          ARRY(24)= EXP(-(0.287+0.52*TMP+0.04475*TMP*TMP))
          IF(MOCAL .NE. 0) GO TO 31
          RETURN
   C      NMCSSC SIOAC DOSE CALCULATIONS
   31     CONTINUE
          TMP = ARRY(23)**(-0.2)
          DO 32 J = 1,5
          JST = 24 + J
          BT = BHRS(J) - TWPN
          BTT = BT - ARRY(23)
          IF(BTT .GE. 0.) GO TO 33
          ARRY( JST) = 0.
          GO TO 32
   33     CONTINUE
          ZZ = 0.5 + 4.5*EXP(-(0.00061 + 0.00025*TMP)*(BTT))
          ARRY(JST) =(TMP - (BT**(-0.2)))*ZZ
   32     CONTINUE
          IF(MOCAL .NE. 1) GO TO 34
          RETURN
   34     CONTINUE
          TMP = 0.
          DO 35 K = 1,5
          KLK = 24 + K
          IF( TMP .GT. ARRY(KLK)) GO TO 35
          TMP = ARRY(KLK)
```

```
      KVL = KLK
   35 CONTINUE
      IF(KVL .NE. 25) GO TO 36
      ARRY(30) = ARRY(25)
      ARRY(31) = BHRS(1)
      GO TO 39
   36 CONTINUE
      IF(KVL .NE. 29) GO TO 37
      ARRY(30) = ARRY(29)
      ARRY(31) = BHRS(5)
      GO TO 39
   37 CONTINUE
      YM = ARRY(KVL - 1)
      YO = ARRY(KVL)
      YP = ARRY(KVL + 1)
      DELLT =  0.25*(ALOG(BHRS(5)) - ALOG(BHRS(1)))
      TO = ALOG(BHRS(KVL - 24 ))
      DELP = YP - YO
      DELM = YO - YM
      DELSQ = DELP - DELM
      DELYO = 0.5*(DELP + DELM)
      OT = - DELYO*DELLT/DELSQ
      XLT = TO + OT
      ARRY(31) = EXP(XLT)
      ARRY(30) = YO -  0.5*DELYO*DELYO/DELSQ
   39 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FALLCB
C     NEVUNS STANDARD.
C     LAST REVISION ON OCT. 27, 1972.
C     TO COMPUTE CROSSWIND DISTANCE EFFECTS FOR THE WSEG 10 -NAS
C     FALLOUT MODEL AND PRODUCE FINAL ANSWERS.
C     CWD IS CROSSWIND DISTANCE IN STATUTE MILES, MDCAL OF 0-ONLY WSEG
C     BIO DOSE, 1-NMCSSC TIME DOSES, 2-ALSO MAX DOSE, ARRY IS A STORAGE
C     ARRAY.
C     FOR OUTPUT THE H + I DOSE RATE IS IN ARRY(32), THE WSEG
C     BIOLOGICAL DOSE IS IN ARRY(33), THE TIME OF FALLOUT ARRIVAL
C     AFTER WEAPON BURST TIME IS IN ARRY(23).
C     THE NMCSSC BIO DOSE AFTER 7,22,68,511, AND 800 HOURS FROM THE
C     STRAT OF THE TIME AXIS IS IN ARRY(34) TO ARRY(38). THE MAX
C     BIOLOGICAL DOSE IS IN ARRY(39) AND THE TIME OF MAX DOSE AFTER
C     TIME ORIGIN IS IN ARRY(31).
C     THE VALUES IN ARRY(32) TO ARRY(39) ARE COMPUTED HERE.

      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWD,MDCAL,TWPN,CWD,ARRY(40)
      TMP = CWD * ARRY(21)*CWD
      IF(TMP .GT. 30.) GO TO 6
      FC = ARRY(20) *EXP(-TMP)
      ARRY (32) = FC*ARRY(22)
    7 CONTINUE
      ARRY(33)= ARRY(32)*ARRY(24)
      IF( MDCAL .NE. 0) GO TO 3
      RETURN
C     NMCSSC SIDAC DOSE CALCULATIONS
    3 CONTINUE
      DO 4 J = 1,5
      ARRY(J + 33) = ARRY(32)*ARRY(J + 24)
    4 CONTINUE
      IF(MDCAL .NE. 1) GO TO 5
      RETURN
    5 CONTINUE
      ARRY(39) = ARRY(30)*ARRY(32)
      RETURN
    6 CONTINUE
      ARRY(32) = 0.
      GO TO 7
      END
```

Subroutines FALL YQ (DECK #20)
FALL WQ (DECK #21)
FALL DQ (DECK #22)
FALL CQ (DECK #23)

A. General

These subroutines represent a fit to the results of the WSEG 10/NAS Biological Dose calculations. They are used in a fashion identical to the basic subroutine except that the parameters MDCAL and TWPC are dummy parameters since only the WSEG Biological Dose is calculated. ~~Thus~~ They are called in order to calculate parameters dpendent on yield, fission fraction~~s~~ and height of burst, on fallout wind and shear, on downwind distance, and on crosswind distance. The most ~~effort in~~ CAREFUL fit~~ting~~ is for appreciable downwind distances and wind velocities, where THE MAXIMUM errors ~~under~~ ARE about 25 percent. ~~occur~~. Near ground zero the calculations are more complex and greater errors may occur. The range of the fit made is generally for yields from 0.2 to 30 MT, winds from 0 to 80 mph, and shear from 0 to 1.6.

The results are given in element 13 of the array ARRYFW which appears in the common block FLWSEG.

B. Requirements on Calling Program

The requirements on the calling program are the same as with the regular WSEG 10/NAS subroutines. The calling program is required to manage the block common FLWSEG and insure the proper data are in the array FLWSEG.

C. Algorithm Implemented

The prime fit in for values of wind greater than 3 mph and OF scaled distance x (downwind distance/wind velocity greater than 3). A look at the WSEG model for large values of scaled distance suggests that the downwind dose factor Fd can be approximated by an equation of the form

$$\log_{10}(Fd) = FAC\,(\alpha + \beta x)\ [= \text{FQWSEG}(11)]$$

For $x < 15.6$, a correction factor equal to $.0015(x-15.6)^2$ has empirically been added to $\log_{10}(Fd)$.

The parameter α is given by

$$\alpha = \alpha_Y + \alpha_W\ [= \text{FQWSEG}(8)],$$

where $\alpha_Y = 5.495 - 0.1099 \log_{10}(\text{Yield})$

$$+ 0.018[\log_{10}(\text{Yield})]^2\ [= \text{FQWSEG}(1)],$$

$\alpha_W = -0.995 \log_{10}(W)$ where W = wind speed.

β is given by

$$\cancel{\beta = \beta_Y \cdot x}$$

~~where~~

$$\beta_{\cancel{Y}} = -0.0641 + .0139 \cdot \log_{10}(\text{Yield}) - 0.0033(\log_{10}(\text{Yield}))^2 .$$

FAC is computed exactly as in subroutine FALLYB [= FQWSEG(5)].

The Biological Dose is given by

$$D = F_d \cdot F_c \cdot FAC\ [= \text{FQWSEG}(13)],$$

where

$$Fc = \frac{1}{\sqrt{2\pi}\ \sigma_c} e^{-1/2\left(\frac{c}{\sigma_c}\right)^2},$$

AND ~~where~~

c is crosswind distance.

~~Again,~~ Inspection of the basic model suggests σ_c is a linear function of x.

We have

$$\sigma_c = A + B \frac{S_c D}{W}$$

where

S_c is the shear,

D is the downwind distance,

$$A = 2 + 1.7309 \log_{10}(\text{Yield}) + 1.2691(\log_{10}(\text{Yield}))^2 \left[= \text{FQWSEG}(3)\right]$$

$$B = 7.55 + 1.8714 \log_{10}(\text{Yield}) - 0.3314(\log_{10}(\text{Yield}))^2 = \left[\text{FQWSEG}(4)\right].$$

It may be noticed that yield dependent parameters in F_d and σ_c are filled by a quadratic function of the yield. This fit is applicable from somewhat under 1 MT to 30 MT.

For scaled distances less than 1 but winds greater than 3 mph the fit is in two parts.

Let $K = 2 - \log_{10}(\text{Yield})$

$$\delta = 3 + 5.6 \log_{10}(W/20)$$

$$L_{MX} = 3.355 - .386 \log_{10}(W/20) - 0.275\, S_c + 0.448(K-1)$$

If $D > K\delta$ then $F_d = 2\pi\sigma_c \cdot L_{MX}$

If $D < K\delta$ then $F_d = L_{MX} - 0.169(D - \delta K)^2$

The calculation for F_c, FAC, and σ_c remain~~s~~ the same.

If the wind speed is less than 3 mph, the following is used to calculate F_d and σ_c

$$\sigma_c = \sigma_A + \sigma_B D$$

where

$$\sigma_A = 3.14 + 0.51 \cdot \text{Yield} - (0.33 + 0.03 \cdot \text{Yield})W$$

$$+ [42.35 - (19.0975 + 0.9225 \cdot \text{Yield})W]\, S_c$$

$$+ [69 - (27.35 + 1.15 \cdot \text{Yield})W]\, S_c^{\,2} \; [= \text{FQWSEG}(18)],$$

$$\sigma_B = (3.611 + 0.039 \cdot \text{Yield})\, S_c \, [= \text{FQWSEG}(19)] .$$

If D > 0 then

$$\log_{10} F_d = A_s + B_s D + C_s D^2$$

where

$$A_s = (4.545 - 0.745 \cdot \text{Yield}) + (0.1222 + 0.0078 \cdot \text{Yield})(1 - W/2)$$

$$- (1.2223 + 0.0278 \cdot \text{Yield})\, S_c$$

$$B_s = -0.06486 + 0.00316 \cdot \text{Yield}$$

$$C_s = (0.2444 - 0.0244 \cdot \text{Yield}) - (0.8977 + 0.1323 \cdot \text{Yield})(1 - W/2) \cdot 10^{-3}$$

The fits for A_s, B_s, and C_s are strictly empirical.

IF $D < 0$ THEN

$$LM_X = 4.35 - 0.56 \log_{10}(\text{YIELD}) - .12W - .15S_c$$

$$\log_{10} F_d = LM_X - \frac{\left(\frac{W}{2} - X\right)^2}{67 + 257 \log_{10}(\text{YIELD})} .$$

```
      SUBROUTINE FALLYQ
C     NEVUNS STANOAROIZED
C     LAST REVISEO NOV. 2, 1972
C     A RAPID VERSION OF THE WSEG 10 FALLOUT MOOEL BASEO UPON FITS
C     TO CALCULATED DOSES RESULTS ARE STOREO IN THE ARRAY QARRY . THE
C     BASIC ARRAY VARIABLES ARE
C     1...YIELO DEPENOENT ALPHA FACTOR IN FD
C     2...YIELO OEPENOENT BETA FACTOR IN FD
C     3...A IN SIGC CALCULATION
C     4...B IN SIGC CALCULATION
C     5...FINAL HOB FACTOR
C     6...IF 0 VARIABLES IN INTERPOLATION RANGE, IF 1 THEY ARE NOT.
C     7...VALUE OF WINO
C     8...ALPHA FACTOR IN FD
C     9...SHEAR/WINO
C    10...2.SIGC.SIGC
C    11...FD
C    12...(1/(SQRT(2.PI)SIGC)) .FD
C    13...WSEG BIOLOGICAL OOSE
C    14 - 19 .. USEO IN SMALL WIND OR DISTANCE CALCULATIONS
C     CALLING SEQUENCE IS SAME AS WITH THE REGULAR MOOEL EXCEPT FOR
C     NMCSSC DOSE OPTION
C     YIELO DEPENDENT CALCULATIONS
      COMMON/FLWSEG/YIELO,FISS,HOB,EFW,SC,OWO,ZILCHA,ZILCHB,CWD,
     1 QARRY(40)
      QARRY(14) = YIELO
      QARRY(6) = 0.
      IF( YIELD .LT. 0.1 .OR. YIELD .GT. 30.) QARRY(6) = 1.
      XLNY = ALOG10(YIELO)
      TMP = XLNY*XLNY
      QARRY(1) = 5.495- 0.1099*XLNY + 0.018*TMP
      QARRY(2) =-0.0641 + 0.0139*XLNY - 0.0033*TMP
      QARRY(3) = 2. + 1.7309*XLNY + 1.2691*TMP
      QARRY(4) = 7.55 + 1.8714*XLNY - 0.3314*TMP
      QARRY(5) = FISS
C     HEIGHT OF BURST CALCULATIONS
      IF(HOB .GT. 0.) GO TO 5
      RETURN
    5 CONTINUE
      XMHB=180.*(YIELO*1000.)**0.4
      IF(HOB.LE.XMHB) GO TO 10
      QARRY(5) = 0.
      RETURN
   10 CONTINUE
      TEMP=HOB/XMHB
      AF=0.5*(1.-TEMP)*(1.-TEMP)*(2.+TEMP)+0.001*TEMP
      QARRY(5) = FISS * AF
```

```
      RETURN
      END
```

```
      SUBROUTINE FALLWQ
C     NEVUNS STANDARDIZED
C     LAST REVISED NOV. 2, 1972
C     WIND DEPENDENT CALCULATIONS
      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWD,ZILCHA,ZILCHB,CWD,
     1 QARRY(40)
      QARRY(7) = EFW
      IF(EFW.LT. 3.) GO TO 5
      QARRY(9) = SC/EFW
      QARRY(8) = QARRY(1) - 0.995*ALOG10(EFW)
      RETURN
    5 CONTINUE
      QARRY(9) = SC
      QARRY(6) = 2.
      WHS = 1. - 0.5*EFW
      QARRY(15) = (4.545 - 0.0745*QARRY(14)) + (0.1222 + 0.0078*
     1 QARRY(14) )*WHS - (1.2222 + 0.0278*QARRY(14) ) *SC
      QARRY(16) = -0.06486 + 0.00316*QARRY(14)
      QARRY(17)=( 0.2444 - 0.02444*QARRY(14) -(0.8977+ 0.1323*QARRY(14))
     1 *WHS )*0.001
      QARRY(18) = 3.14 + 0.51*QARRY(14) - (0.33+ 0.03*QARRY(14))*EFW
     1 + (42.35+ (-19.0975 + 0.9225*QARRY(14))*EFW)*SC
     2 + (69. + (-27.35  + 1.15 *QARRY(14))*EFW)*SC*SC
      QARRY(19) = (3.611 + 0.039*QARRY(14) )*SC
      RETURN
      END
```

```
      SUBROUTINE FALLDQ
C     NEVUNS STANDAROIZED
C     LAST REVISED NOV. 2, 1972
C     OOWNWIND OISTANCE DEPENDENT CALCULATIONS
      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWO,ZILCHA,ZILCHB,CWD,
     1 QARRY(40)
                          2
      IF( QARRY(7) .LT. 1.) GO TO 20
      XSCL = DWO/QARRY(7)
      SIGC = QARRY(3) + QARRY(4)*QARRY(9)*DWO
      IF(XSCL .LT . 1.) GO TO 5
C     NORMAL WIND SPEED AND DISTANCE FIT
      XLGFD = QARRY(8) + QARRY(2)*XSCL
C     TUSE  SUPRESSES THE SIGC FACTOR ON FO FOR FITS A! SMALL
C     DISTANCES OR LOW WIND SPEEDS WHICH WERE DIRECTLY ON DOSE.
      TUSE = 2.5066*SIGC
      IF(XSCL .GT. 15.6) GO TO 10
C     CORRECTION FOR SCALED DISTANCES UNOER 15.6.
      TMP = XSCL - 15.6
      XLGFD = XLGFD + 0.0015*TMP*TMP
   10 CONTINUE
      QARRY(10) = 2.*SIGC*SIGC
      IF( XLGFD .LT. -8.) XLGFO = -8.
      QARRY(11) = QARRY(5) * 10.**XLGFD*QARRY(14)
      QARRY(12) = QARRY(11)/TUSE
      RETURN
C     FIT FOR SMALL DISTANCES ANO NORMAL WINO SPEEDS.
    5 CONTINUE
      XLW = ALOG10(QARRY(7)/20.)
      DELTN = 3. + 5.6*XLW
      FACT = 2. - ALOG10(QARRY(14))
      DELTA = DELTN*FACT
      XLOMXN = 3.355 - 0.386*XLW - 0.275*QARRY(9)/QARRY(7)
      XLOMX = XLOMXN + 0.448*(FACT - 1.)
      OWDS = OWO*FACT
      IF ( DWOS .LT. DELTA) GO TO 6
      XLGFO = XLOMX
      TUSE = 1.
      GO TO 10
    6 CONTINUE
      XLGFD = XLOMX- (DWOS-OELTA)*(OWOS-OELTA)*0.0169
      GO TO 10   TUSE=1
C     FIT FOR LOW WIND SPEEDS.
   20 CONTINUE
      IF( OWD .GT.0. )GO TO 21
C     DISTANCE LESS THAN ZERO
      XLY = ALOG10(QARRY(14))
      XLOMX = 4.35 - 0.56*XLY      - 0.12*QARRY(7)-0.15*QARRY( 9)
      BDT = 67. + 257.*ALOG10(QARRY(14)   )
      DUSE = 0.5*QARRY(7) - DWD
```

```
            XLGFD = XLDMX - DUSE*DUSE/BOT
            GO TO 22
      C     DISTANCE GREATER THAN ZERO
      21    CONTINUE
            XLGFD = QARRY(15) + QARRY(16)*DWD + QARRY(17)*DWD*DWD
            IF(QARRY(17) .LT. 1.E-8) GO TO 22
            BETA = -1.*QARRY(16)/(2.*QARRY(17))
            IF( DWD .GT. BETA) XLGFD = -8.
      22    CONTINUE
            SIGC = QARRY(18) + QARRY(19)*DWD
            TUSE = 1.
            GO TO 10
            END
```

```
      SUBROUTINE FALLCQ
C     NEVUNS STANDARDIZED
C     LAST REVISED NOV. 2, 1972
C     CROSSWIND DISTANCE DEPENDENT CALCULATIONS
      COMMON/FLWSEG/YIELD,FISS,HOB,EFW,SC,DWD,ZILCHA,ZILCHB,CWD,
     1 QARRY(40)
      TMP = CWD*CWD/QARRY(10)
      IF(TMP .GT.10.) TMP = 10.
      QARRY(13) = QARRY(12)*EXP(-TMP)
      RETURN
      END
```

SUBROUTINE CFALLY(Deck #211)

A. GENERAL

This subroutine and CFALWD give the cluster model for fallout. It is used to determine overall fallout dose from a group of weapons closely spaced. It is based on WSEG-10 model and uses the equations given in P-1065 "Methodology of Fallout Risk Assessment." This subroutine computes yield-dependent parameters and stores them in the array STR(5). The subroutine CFALWD uses, in addition, the wind, wind shear, downwind distance, and crosswind distance to compute the maximum.

B. REQUIREMENTS ON CALLING PROGRAM

Communication is through the block common /CLFLPR/. The parameters of interest to this subprogram are

Input: YIELD--weapon yield in megatons
FISS--Weapon fission fraction

Output: STR(5)--derived values used as input to subprogram CFALWD.

C. ALGORITHM IMPLEMENTED

Five output values are directly computed. Call Y the yield. Then,

$$STR(1) = 7.5 + 1.66 \log_{10} Y$$

$$STR(2) = 2.71/STR(1)^{1.382}$$

$$STR(3) = 2. + 3 \log_{10} Y$$

$$STR(4) = 7.5 + 1.5 \log_{10} Y$$

$$STR(5) = 2 \times 10^{6} \times Y \times FISS .$$

```
      SUBROUTINE CFALLY

C     CALCULATES FALLOUT DOSES FROM CLUSTERS OF WEAPONS USING A
C     SIMPLFIED WSEG 10 FALLOUT MODEL.
C     XL IS THE HALF LENGTH OF THE CLUSTER IN THE DOWNWIND DIRECTION
C     SIGW IS STD. DEV. OF CLUSTER IN TH E CROSSWIND DIRECTION.
C     DWN DOWNWIND DIRECTION,CRS CROSSWIND DIRECTION.
C     STR IS USED TO STORE YIELD DEPENDENT PARAMETERS
C     STR(1) IS CHARACTERISTIC TIME T.
C     STR(2)  IS 2.71/T**1.382
C     STR(3)  IS A USED TO CALCULATE SHEAR SIGMA
C     STR(4) IS B USED TO CALCULATE SHEAR SIGMA
C     STR(5) IS PRODUCT OF YIELD TIMES FISSION FRACTION TIMES K.
C     ASSUMES 0 HEIGHT OF BURST SO ANY CORRECTION MUST BE EXTERNAL.

      COMMON/CLFLPH/YIELD,FISS,WIND,SHR,XL,SIGW,DWN,CRS,DOSE,STR(5)

      ALT = ALOG10(YIELD)
      STR(1) = 7.5 + 1.66*ALT
      STR(2) = 2.71/(STR(1)**1.382)
      STR(3) = 2. + 3.*ALT
      STR(4) = 7.5 + 1.5*ALT
      STR(5) = 2400000.*YIELD*FISS
      RETURN
      END
```


PRECEDING PAGE BLANK-NOT FILMED

SUBROUTINE CFALWD(Deck #212)

A. GENERAL

This is the second subroutine, following CFALLY, to implement the cluster fallout model. This subroutine has as input yield-dependent parameters computed in CFALLY, wind and distance, and outputs the WSEG-10 Biological Dose.

B. REQUIREMENTS ON CALLING PROGRAM

Communication is through the common block /CFALWD/. Parameters of interest are

Input: STR(5)--yield-dependent parameters which must be computed by a call to CFALLY

WIND--wind speed, mph

SHR--wind shear, mph/kilofoot

XL--half the downwind extent of the cluster

SIGW--the crosswind standard devaition of the fission yield of the cluster weapons

DWN--downwind distance, miles

CRR--crosswind distance, miles

Output: DOSE--computed WSEG-10 biological dose

C. ALGORITHM IMPLEMENTED

The calculations are a direct implementation of the equations in IDA Paper P-1065. First the crosswind shear standard deviation is computed as SIGC = STR(3) + STR(4)(DWN+XL) · SHR/WIND. Now a downwind contribution to dose, FD, is computed. If the downwind distance is greater than cluster distance (DWN > XL), then: RAT = DWN/(WIND · STR(1)). For RAT < 0.6, an explicit calculation is done as FD = STR(2) · exp(-RAT)/(WIND · RAT$^{.382}$). For RAT $\geq$ 0.6 a polynomial approximation to the above expression is

PRECEDING PAGE BLANK-NOT FILMED

used by FD = STR(2)/(WIND · FAC^4), where FAC = 0.85419 + .421 RAT $-0.0019286\ RAT^2 + 0.00929\ RAT^3$. If DWN < -XL the dose is set to 0 and the subprogram exited. Otherwise, the factor FD is computed by FD = STR(2)(DWN+XL) exp(-RAT)/(2 · WIND · XL · $RAT^{0.382}$), where RAT = XL/(WIND/STR(1)). This gives a linear buildup of dose from the upwind edge to the downwind edge of the cluster.

Now the crosswind factor, FC, is computed. For long distances (DWN $\geq$ 5 · SIGW), a normal distribution is used. First a standard deviation is computed by

$$SIGU = \sqrt{SIGC^2+SIGW^2} \ .$$

Now a value TMP is computed by

$$TMP = 0.5CRS^2/SIGU^2 \ .$$

If TMP > 6 the dose is set to 0 and the subprogram exited. Otherwise,

$$FC = \left(1/\sqrt{2\pi}\right) \exp(-TMP)/SIGU.$$

For close distances the crosswind dose is the difference between two cumulative normal functions:

$$FC = \frac{0.5}{\sqrt{2}\ SIGW}\left(\text{cumnor}\left(\frac{|CRS|}{SIGC} + \frac{\sqrt{2}\ SIGW}{SIGC}\right) - \text{cumnor}\left(\frac{|CRS|}{SIGC} - \frac{\sqrt{2}\ SIGW}{SIGC}\right)\right) .$$

Finally, the dose is computed by

$$DOSE = STR(5) \cdot FD \cdot FC \ .$$

```
      SUBROUTINE CFALWD
C     FOR WEAPONS IN CLUSTERS
C     DOES BOTH WIND DEPENDENT AND DISTANCE DEPENDENT CALCULATIONS.
C     ASSUMES CFALLY HAS BEEN CALLED TO FILL STR
C     CLUMSY WRITING USED TO ATTEMPT TO SPEED CALCULATION.
      COMMON/CLFLPR/YIELD,FISS,WIND,SHR,XL,SIGW,DWN,CRS,DOSE,STR(5)
      SIGC = STR(3) + STR(4)*(DWN + XL)*SHR/WIND
      IF(DWN .LT.XL) GO TO 21
C     HERE WE ARE BEYOND THE CLUSTER
      RAT = DWN/(WIND*STR(1))
      IF(RAT .LT. 0.6) GO TO 11
      IF(RAT.GT.8.)GO TO 23
C     USE POLYNOMIAL APPROXIMATION HERE.
      FAC = 0.85419 + RAT*(0.421 + RAT*(-0.0019286 + RAT*0.00929))
      FACS = FAC*FAC
      FO = STR(2)/(WIND*FACS*FACS)
12    CONTINUE
      IF(DWN .LT. 5.*SIGW) GO TO 15
C     SOMETHING LIKE DWN .LT. 75. COULD ALSO BE USED HERE
      SIGUS = SIGC*SIGC + SIGW*SIGW
C     SIGUS IS SHR SIGMA MODIFIED FOR CLUSTER SIZE FOR DOWNWIND APPRX.
      SIGU = SQRT(SIGUS)
      TMP = 0.5*CRS*CRS/SIGUS
      IF(TMP .GT. 6.) GO TO 23
      FC = 0.39894228*EXP(-TMP) /SIGU
18    CONTINUE
      DOSE = STR(5)*FO*FC
      RETURN
15    CONTINUE
C     COMPUTE CROSSWIND FACTOR HERE BY DIFFERENCE OF TWO CUMULATIVE
C     NORMALS. CUMNOR APPROXIMATION DIRECT FROM HASTINGS.
      R = 1.414*SIGW
      TMP1 = ABS(CRS)/SIGC
      TMP2 = R/SIGC
      X = TMP1 + TMP2
      XX= TMP1 - TMP2
      IF(XX .GT. 6.) GO TO 23
       IF(X.LT.4.0) GO TO 31
       IF(XX.LT.4.0) GO TO 32
      FC=0.0
      GO TO 18
  32   CONTINUE
      IF(XX.GT.-4.0) GO TO 33
      FC=0.5/R
      GO TO 18
 33   CONTINUE
      CMN 1=1.0
  31   CONTINUE
      FAC = 1. + X*(0.278393+X*(0.230389+X*(0.00972+X* 0.078108)))
      FACS = FAC * FAC
      CMN1 = 1. - 0.5/(FACS*FACS)
      IF(XX.LT.4.0) GO TO 35
      CMN2=1.
      GO TO 17
```

```
 35   CONTINUE
      IF(XX.GT.-4.0)    GO TO 36
      CMN2=0.0
      GO TO 17
      CONTINUE
      X = ABS(XX)
      FAC = 1. + X*(0.278393+X*(0.230389+X*(0.00972+X* 0.078108)))
      FACS = FAC * FAC
      IF(XX.LT. 0.) GO TO 16
      CMN2 = 1. - 0.5/(FACS*FACS)
      GO TO 17
16    CONTINUE
      CMN2 = 0.5/(FACS*FACS)
17    CONTINUE
      FC = 0.5*(CMN1 - CMN2)/R
      GO TO 18

11    CONTINUE
C     USE EXPLICIT CALCULATION RATHER THAN APPROXIMATION BELOW
      FO = STR(2) *EXP(-RAT)/(WINO*RAT**0.382)
      GO TO 12


21    CONTINUE
C     HERE WE ARE IN THE CLUSTER AND USE LINEAR VARIATION OF FO ALONG IT
      IF(OWN .GT.-XL) GO TO 22
C     UPWINO OF CLUSTER
23    CONTINUE
      OOSE = 0.
      RETURN
2     CONTINUE
      RAT = XL/(WIND*STR(I))
      IF(RAT.GT.8.0) GO TO 23
      FD = STR(2)*(OWN + XL)*EXP(-RAT)/(WIND*2.*XL*RAT**0.382)
      GO TO 15
      END
```

SUBROUTINE LSCRN (DECK #24)

A. GENERAL

This subroutine provides the maximum distances at which certain RADIOACTIVE FALLOUT doses occur. As such, it can be considered as the inverse of the WSEG-10 routines FALLYB, etc., which compute doses given distance. This subroutine is intended for screening calculations to determine the limits to where weapons must be considered for depositing significant dosage upon a target area. For this reason the equations are rather simple since speed of calculation is competitive in importance with accuracy. Nonetheless, at dose levels of .1 to 10 R (based on a fission fraction of 1) and wind velocities over 10 mph the error in distance calculation is usually under 20 percent.

The subroutine provides maximum downwind distance for the specified dose level, maximum crosswind distance, downwind distance at which the maximum crosswind distance occurs, and maximum upwind distance. These are given in elements 7, 8, 6, and 9, respectively, of the array SCARY in the common block ~~RVSCRN~~ FLSCRN used to communicate with the subroutine. These values could be used to construct elliptical contours which would approximate the actual contour shaped.

B. REQUIREMENTS ON CALLING PROGRAM

The calling program is responsible for managing the common block FLSCRN since many calls may be made on this subroutine and such management may reduce calling times. These values to be supplied are:

JDO	a control parameter
YLDS	weapon yield in MT
FISSS	weapon fission fraction (may be larger than one to represent several closely located weapons)
HOBS	weapon height of burst (ft)
WDNS	fallout wind speed (mph)
SHRS	shear in mph/Kft
DOSES	the WSEG biological dose in rads for which distances are to be found

If the parameter JDO is negative only the yield dependent parameters are calculated and the subroutine is exited. If the parameter JDO is positive it is assumed the yield dependent parameters are in elements 1 and 2 of the array SCARY and the calculation proceeds. The calling program is responsible to insure that these parameters are in fact supplied. If JDO is one, only maximum downwind distance is supplied in element 7 of the array SCARY. If it is greater than one, the maximum crosswind distance is also supplied in element 8, downwind distance ~~to~~ for maximum crosswind distance in element 6, and maximum upwind distance in element 9 of the array.

The elements 3, 4, and 5 are not used in this subroutine in order to provide compatibility in input and those answers in elements 7 to 9 between this subroutine and the subroutine RSCRN.

C. ALGORITHM IMPLEMENTED

The dose is adjusted for height of burst effects by the same equations as in the subroutine FALLYB, since the fit is made for height of burst of zero. If the height of burst is above the maximum for significant effects, distances of zero are returned.

If the wind is less than 2 mph the wind is forced to be 2 mph and the same distances are used upwind as downwind.

For doses less than 10 rad the maximum downwind distance is computed by

$$D = W[x_o - 15 \cdot A + (0.55 + 0.75 \cdot S_c + 0.0075\ W)A^2]$$

where

W = wind speed

S_c = wind shear

D = max downwind distance

$$x_o = \begin{cases} 60.8516 - 36.5944\ S_c + 13.1363\ S_c^2 - 14.8\ \log_{10}(W), & S_c < 1.3929 \\ 21.5659, & S_c \geq 1.3929 \end{cases}$$

$A = \log_{10}(\text{Dose}) - \log_{10}(\text{yield})$

For doses above 10 rad the maximum downwind distance is computed by

$$D = W[\bar{x}_o - 10\cdot A + (1 + .75\ S_c + 0.0075W)A^2]$$

where

S_c is limited here to a maximum value of 0.6

$\bar{x}_o = 16.1 + 26.2(S_c - 0.6)^2 - 8.1\ \log_{10}(W)$

For doses less than 100 rad the maximum crosswind distance is computed by

$$C = 160S_c \cdot \log_{10}(10\ \text{Yield}) \cdot (2.2041 - \log_{10}(\text{Dose}) - 1.18(4 - \log_{10}(\text{Dose}) \cdot \log_{10}(W/10))).$$

The distance of maximum crosswind is computed by

$$D_c = (80 + 18.7 - 7.5\ \log_{10}(\text{Dose}) - 7.5 \cdot S_c)W(1 + 0.75\ \log_{10}(\text{Yield})).$$

The maximum upwind distance is given by

$$D_u = \log_{10}(10 \cdot \text{yield}) \cdot (6.5 - \log_{10}(\text{Dose}) - 1.25 \cdot \log_{10} W).$$

For doses greater than 100 we have:

For maximum crosswind distance

$$C = 2.5\ S_c\ \log_{10}(10 \cdot \text{Yield})(50 - 20\ \log_{10}(\text{Dose}/100) - (17.5 - 2.5\ \log_{10}(\text{Dose}/100))\log_{10}(W/10))$$

For distance of maximum crosswind distance

$$D_c = \frac{W}{20} \cdot \{0.5\ \log_{10}(10 \cdot \text{Yield})[130 - 60\ \log_{10}(\text{Dose}/100) + (500 - 250\ \log_{10}(\text{Dose}/100)) \cdot \log_{10}(W/10)]\}$$

For maximum upwind distance

$$D_u = 2 + \log_{10}(10 \cdot \text{Yield}) \quad .$$

In some cases the error in the fit may give negative distances at very high doses. In these cases the distance is set equal to zero.

```
      SUBROUTINE LSCRN
C     NEVUNS STANDARDIZED
C     LAST REVISEO NOV. 2, 1972


C     BASED ON A FIT TO THE WSEG 10 NAS MODIFIEO FALLOUT MODEL.
C     WSEG BIOLOGICAL OOSE OCCURS. IT IAN BE USED FOR SCREENING
C     OISTANT WEAPONS.
C     THE ARRAY SARRY IS OIMENSIONEO 9 ANO SUPPLIEO BY THE CALLING
C     PROGRAM. IF JDO IS POSITIVE IT IS ASSUMED THE YIELD DEPENDENT
C     PARAMETER IS ALREAOY SUPPLIEO.  IF JDO HAS MAGNITUDE ONE ONLY
C     THE OOWNWINO OISTANCE IS COMPUTED. IF MAGNITUOE GREATER THAN ONE
C     THEN ALL OISTANCES ARE COMPUTED. THE ANSWERS ARE IN THE ARRAY
C     SARRY
C     FILL YIELD DEPENDENT PARAMETERS WITH JOD =0,NO ANSWERS THEN.
C....SARRY(7) -- MAXIMUM DOWNWIND DISTANCE
C... SARRY(8) -- MAXIMUM CROSSWIND OISTANCE
C... SARRY(6) -- OISTANCE AT WHICH MAXIMUM CROSSWIND OCCURS.
C... SARRY(9) -- MAXIMUM UPWIND DISTANCE


      COMMON/FLSCRN/JDO,YIELD,FISS,HOB,WND,SHR,ODSE,SARRY(9)

      IF (JDO .GT. 0) GO TO 10
C     YIELD DEPENDENT CALCULATIONS
      SARRY(1) = ALOG10(YIELD) + 1.
      SARRY(2) = 180.*(YIELD*1000.)**0.4
      RETURN
   10 CONTINUE

C     NON YIELD DEPENDENT CALCULATIONS
      OISE = DOSE
      VNO = WND
      SHA = SHR
      IF( HOB .EQ. 0.) GO TO 21
      IF( HOB .LE.(SARRY(2) - 0.01))GO TO  22
C     HOB TOO HIGH FOR ANY FALLOUT. SET ALL DISTANCES TO ZERO.
      SARRY(6) = 0.
      SARRY(7) = 0.
      SARRY(8) = 0.
      SARRY(9) = 0.
      RETURN
22    CONTINUE
C     HOB DOSE ADJUSTMENT
      SHOB = HOB/SARRY(2)
      AF=0.5*(1.-SHOB)*(1.-SHOB)*(2.+SHOB)+0.001*SHOB
      OISE = OISE /AF
21    CONTINUE
      DISE = DISE/FISS
      KDO = 0
      IF( WND .GT. 2.) GO TO 5
C     FORCE ALL WINDS TO AT LEAST 2 FOR SCREENING CALCULATIONS
```

```
              VND = 5.
        C     GIVE SAME DOSES FOR NEGATIVE DISTANCES FOR SMALL WINDS
              KOD = 1
        5     CONTINUE
              IF( DISE .GT. 10.) GO TO 31
        C     LOW DOSEAGE FIT
              XLWN = ALOG10(VND)
              IF( SHA .GT. 1.3929) GO TO 12
              XSWO = 60.8516 - 36.5944*SHA + 13.1363*SHA*SHA -14.8*XLWN
         13   CONTINUE
              XLGD = ALOG10(DISE)
              XLG = XLGD - SARRY(1) + 1.
              XSW = XSWO - 15.*XLG +(0.55 + 0.75*SHA + 0.0075*VND)*XLG*XLG
        C     MAX DOWNWIND DISTANCE
              SARRY(7) = XSW*VND
        32    CONTINUE
              IF( JDO .GT. 1) GO TO 14
              RETURN
         12   CONTINUE
              XSWO = 21.5659
              GO TO 13

         14   CONTINUE
              IF (DISE .GT. 100.) GO TO 33
              XLGW = XLWN - 1.
        C     CROSSWIND DISTANCE
              SARRY(8) = 160.*SHA* SARRY(1)*(2.2041 -XLGD)- 1.18*(4. -XLGD)
             1 *XLGW
        C     DISTANCE OF MAX CROSSWIND
              SARRY(6) =           (80. +(18.7 - 7.5*(XLGD + SHA))*VND) *(1. +
             1  0.75*(SARRY(1) - 1.))
        C     MAX UPWIND DISTANCE
              SARRY(9) = SARRY(1)*(  6.5 - XLGD - 1.25*XLGW)
              IF( KOD .EQ. 1) SARRY(9) = SARRY(7)
              RETURN


        31    CONTINUE
        C     HIGH DOSE
              IF( SHA .GT. 0.6) SHB = 0.6
              TEMP = SHB - 0.6
              XSWO = 16.1 + 26.2*TEMP*TEMP - 8.1*XLWN
              XLGD = ALOG10(DISE)
              TEMP = XLGD - SARRY(1) - 1.
              XSW = XSWD - 10.*TEMP+ (1. + 0.75*SHB + 0.0075*VND)*TEMP*TEMP
              SARRY(7) = XSW*VND
              IF(SARRY(7) .LT. 0.) SARRY(7)  = 0.
              IF (JDO .GT. 1) GO TO 34
              RETURN
        34    CONTINUE

        33    CONTINUE
              TEMP = XLGD - 2.
              SARRY(8) = 2.5*SHA*SARRY(1)*((50.- 20.*TEMP) - (17.5 -2.5*TEMP)
```

```
            1 *(XLWN - 1.))
              IF (SARRY(8) .LT. 0.)SARRY(8) = 0.
              DSW = 0.5*SARRY(1)*((130.- 60.*TEMP)+(500. - 250.*TEMP)
            1  *(XLWN - 1.) )
115           SARRY(6) = DSW*VND /20.
              IF (SARRY(6) .LT. 0.)SARRY(6) = 0.
              SARRY(9) = SARRY(1) + 2.
              IF( KDO .EQ. 1) SARRY(9) = SARRY(7)
              RETURN
120           END
```

SUBROUTINE RSCRN (DECK #25)

A. GENERAL

This subroutine is used for screening Fallout Distance. It is based upon equations developed by R. Mason of the ~~NMCCS~~ NMCSSC which are implemented in the SIDAC damage assessment model.

The use of this subroutine is almost identical to the subroutine LSCRN. This description shall only indicate differences from the subroutine LSCRN. The major difference in usage is that the downwind distance ~~to~~ FOR the maximum crosswind distance is not ~~implemented~~ CALCULATED in this subroutine. The calculation of screening distance after the yield dependent parameters are determined, is somewhat faster with this subroutine.

B. REQUIREMENTS IN THE CALLING PROGRAM

The requirements are the same except that 6 yield dependent parameters, rather than 2, are computed and stored in elements 1-6 of the array SCARY.

C. ALGORITHMS IMPLEMENTED

The height of burst correction is computed as in the subroutine LSCRN.

The following minimum or maximum values are FORCED

Sc(Wind Shear) $\geq$ 0.05

Y(Yield,MT) $\geq$ 0.001

Dose (corrected for HOB) $\leq$ 3000.

For W (WIND) < 1 set W = 3 and S_c = 0.2
and set upwind equal to downwind distance

For upward distance

$$D = -(-2.56928Y^{.073064} \ln_e (\frac{\text{Dose W } S_c}{40}) + 19.59589Y^{0.13039})^2 \frac{W}{20} .$$

AD-A039 819 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA PROGRAM --ETC F/G 15/4
DOCUMENTATION OF CURRENT IDA COMPUTER MATERIAL DEVELOPED FOR DC--ETC(U)
JAN 77 L A SCHMIDT DCPA01-76-C-0213

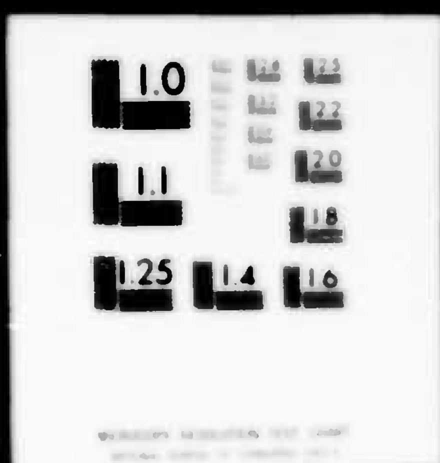
UNCLASSIFIED IDA/HQ-77-19226 NL

3 OF 3
AD A039 819

END
DATE
FILMED
6-77

3 OF 3

ADA039 819



For maximum crosswind distance

$$C = \left(-0.78095Y^{0.12239} \ln_e \left(\frac{\text{Dose } W \cdot S_c}{40}\right) + 5.84722Y^{0.19279}\right)^2 \cdot \frac{S_c}{2} .$$

For maximum upwind distance

$$D_o = (2.76Y^{.24363})^2 \quad .$$

```
      SUBROUTINE [illegible]
C     [illegible] STANDARDIZED
C     LAST REVISED NOV. 8, 197[illegible]

C     SCREENING DISTANCES BY R. NASON AS IN [illegible]
C     BASED ON A FIT TO THE [illegible] IS HAS MODIFIED FALLOUT MODEL.
C     THE ARRAY GAMMY IS SUPPLIED BY THE MAIN PROGRAM. IT HAS DIMENSION
C     9. IF JDO = 1 IT IS ASSUMED THE YIELD DEPENDENT PARAMETERS ARE
C     FILL YIELD DEPENDENT PARAMETERS WITH JDO [illegible] THEN.
C     ALREADY SUPPLIED IN GAMMY 1 - 6. [illegible] ARE IN GAMMY 7 TO 9.
C...  GAMMY(7) IS DOWN WIND DISTANCE AT WHICH THE [illegible] DOSE OCCURS.
C...  GAMMY(8) IS THE MAXIMUM CROSSWIND DISTANCE FOR THIS DOSE
C...  GAMMY(9) IS THE MAXIMUM UPWIND DISTANCE FOR THIS DOSE.

      COMMON/FLDCOM/JDO,YIELD,FISS,HOB,WND,SHR,DOSE,GAMMY(9)

      IF( JDO .EQ. 1) GO TO 1
C     DO YIELD DEPENDENT PARAMETERS.
      IF( YIELD .LT.0.001) YIELD = 0.001
      GAMMY(1) = [illegible]
      GAMMY(2) = [illegible]
      GAMMY(3) = [illegible]
      GAMMY(4) = [illegible]
      TMP = [illegible]
      GAMMY(5) = [illegible]
      GAMMY(6) = [illegible]
      RETURN

    1 CONTINUE
C     NON YIELD DEPENDENT PARAMETERS.
      DISE = DOSE
      WND = WND
      SHR = SHR
      IF( HOB .EQ. 0.) GO TO 21
      IF( HOB .GE. (GAMMY(6) * [illegible])) GO TO 22
C     NO FALLOUT BECAUSE HOB IS TOO HIGH. SET ALL DISTANCES TO ZERO.
      GAMMY(7) = 0.
      GAMMY(8) = 0.
      GAMMY(9) = 0.
      RETURN
   22 CONTINUE
      SHOB = HOB/GAMMY(6)
      [illegible]
      DISE = DISE/[illegible]
   21 CONTINUE
      DISE = DISE/FISS
      JDO = 0
      IF(WND .GT. 3.) GO TO 3
C     FORCE AT LEAST 3 MPH WIND.
      WND = 3.
```

```
      SMA = 0.8
      ADD = 1
    5 CONTINUE
      IF(SMA .LT. 0.05) SMA = 0.05
      IF (DIST .GT. 3000.) DIST = 3000.
      AMP = DIST*V*SMA/60.
      AL = ALOG(AMP)
      TMP = GAMMY(1)*AL + GAMMY(2)
C     DOWNWIND DISTANCE
      GAMMY(7) = TMP*TMP*VWD/20.
      TMP = GAMMY(3)*AL + GAMMY(4)
C     MAXIMUM CROSSWIND DISTANCE
      GAMMY(8) = TMP*TMP*SMA/0.3
C     UPWIND DISTANCE
      GAMMY(9) = GAMMY(8)
      IF (ADD .EQ. 1) GAMMY(9) = GAMMY(7)
      RETURN
      END
```

SUBROUTINE QKINR (Card Deck [illegible])

A. GENERAL

This subroutine is a quick means of calculating three types of Initial Nuclear Radiation Doses: Prompt Gamma, Secondary Gamma, and Neutrons. It is based upon work by L. Spencer and C. Eisenhauer of the National Bureau of Standards. The prompt gamma in their method was based on a calculation involving a somewhat lengthy numerical integration involving fireball rise. The routine has a numerical fit to those results. The calculation for the other two types of doses is the same as in the original calculation.

B. REQUIREMENTS ON CALLING PROGRAM

Communication with the subprogram is through the common block /QINRPR/. The following variables must be defined.

W Weapon yield in kilotons ($0.1 \leq W \leq 30{,}000$).

XLW Logarithm to the base 10 of W.

HB Height of burst in meters (not needed).

HBR Scaled height of burst in hundreds of feet, i.e., $HB/W^{0.33333}$, $0 \leq HBR \leq 13$.

RHOO Air Density in grams/cm^3. Assumed 1.1 in calculating prompt gamma; may be any value for.

ITP Weapon type: 1 = thermonuclear, 2 = intermediate, 3 = fission.

FF Fission fraction.

RO Slant range, meters

CF Collision factor, a value of 2 is usually used.

RBE Relative Neutron Biological Effectiveness, 1 is usually used.

The subroutine returns the following variables

DOSIKE Prompt gamma dose.

DOSGAM Secondary gamma dose.

DOSNEU Neutron dose.

DOSTOT Sum of above three doses.

C. ALGORITHM IMPLEMENTED

1. INTRODUCTION

A computer subroutine for the calculation of the Initial Nuclear Radiation Dose has been developed by C. Eisenhauer and L. Spence of the National Bureau of Standards. This subroutine computes the dose from fission product gamma radiation, secondary gamma radiation, and neutrons. The calculations are based upon a paper by French and Mooney.[1] The calculations for secondary gamma rays and neutrons are made from equations of the form

$$D = a \exp b[\exp(cR_o^2 + dR_o)] ,$$

where R_o is the slant range from weapon to monitor point. This can be implemented for relatively rapid calculation. The calculations for the fission product gamma dose, on the other hand, are much more lengthy. They require more complex expressions for dose as a function of time. These expressions must be integrated as a function of time to obtain the total dose.[2] As a result the computer routine which implements these equations is a relatively slow calculation. For analysis where the doses must be calculated a large number of times, in particular in damage assessment calculations involving many weapons and monitor points, such long running calculations can add most substantially to computer requirements.

An algorithm for a more rapid calculation of the fission product gamma dose is described below. This algorithm is a strictly numerical fit to the results of calculations of fission product gamma doses over a range of parametric values. The fit is generally within 10 to 20 percent over the range of interest. This range is for yields, W, ranging from 0.1 to 10,000 KT, scaled heights of burst (the height of burst/$W^{1/3}$) from 0 to 1,300 feet,

---

[1] R. L. French and L. G. Mooney, "Initial Radiation Exposure from Nuclear Weapons," Radiation Research Associates, Inc., Interim Report on OCD Contract No. DAHC20-72-C-0123, RRA-T7201, 15 July 1972.

[2] The essential reason for the integration is due to the hydrodynamic buoyancy of the fireball which not only changes the distance between source and receiver, but due to changes in air density with height changes the radiation adsorption in a complex manner.

and slant ranges ranging from either the minimum slant range at ground zero, or the minimum slant range where the fission product gamma dose is less than 1/20 the neutron dose, to a maximum slant range where the fission product gamma dose is about 10R. The algorithm will be described directly first, followed by a few comments concerning its development.

B. ALGORITHM FOR FISSION PRODUCT GAMMA DOSES

A maximum slant range, SRM, is computed by

$$SRM = 1{,}898 + 848.25 \cdot L + 111.241 \cdot L^2 ,$$

where $L = \log_{10}(W)$. If $R_o >$ SRM, the Dose, D, is 0; otherwise, the following procedure is followed.

An "asymptotic logarithmic" dose, D_{asy}, is computed as follows. Let

$$R_z = 1{,}188.13 + 84.3259L + 71.2003L^2 + 12.4084L^3 - 9.7729L^4 + 2.837411L^5 .$$

$$D_{asy} = 7 - (R_o - R_z)/660 .$$

If $H_B > 0.2$, set $D_{asy} = D_{asy} + R_A$,

where H_B = scaled height of burst $((\mathrm{ft}/(\mathrm{KT})^{1/3})/100)$,

$$R_A = -b \cdot H_B + c \cdot H_B^{\;2} ,$$

with $b = 1.24154 \cdot 10^{-2} + 6.0937 \cdot 10^{-3}L + 3.46545 \cdot 10^{-3}L^2 + 1.534 \cdot 10^{-3}L^3 - 5.9337 \cdot 10^{-5}L^4$;

$c = 9.9037 \cdot 10^{-5} - 9.147 \cdot 10^{-5}L + 1.963 \cdot 10^{-4}L^2 + 1.83616 \cdot 10^{-4}L^3 - 1.069 \cdot 10^{-4}L^4 + 1.8162 \cdot 10^{-5}L^5$;

Compute a difference dose, D_p, by the following procedure.

If $L \leq 2.4$:

$$S_B = 362 + 74.3 \cdot L - 55.99 \cdot L^2 + 34.59 \cdot L^3 .$$

For $R_o \geq S_B$, let

$$D_F = D_o + m(x-x_o) \; ,$$

where $x = 100/R_o$;

$D_o = -0.015 - 0.0056H_B$;

$x_o = 0.055 - 0.00135H_B$;

$m = m_o + m_S H_B$.

and for $L < 1$

$m_o = 0.552 + 0.398L + 0.25L^2$;

$m_S = 0.0518 + 0.02985L + 0.01685L^2$;

and for $L \geq 1$

$m_o = -1.71 + 2.78L$;

$m_S = 0.1690 - 0.0615L$.

For $R_o < S_B$, let

$$D_F = D_o + m(x-x_o) + c(x-100/S_B)^2 \; ,$$

where D_o, x_o, and m are computed as before and

$$c = \begin{cases} 0.22 + [0.575(L+1)]^4 \; , & L \leq 1.602 \; , \\ 0.22 + [1.1144(L-0.61)]^4 \; , & L > 1.602 \; . \end{cases}$$

Now if $L > 2.4$, we have:

if $R_o \geq S_B$

$$D_F = m(x-x_o) \; ,$$

where $x = 100/R_o$;

$m = -74.8819 + 95.3347L - 40.2997L^2 + 6.06248L^3$;

$x_o = 0.0217(4.76-L)$.

But if $R_o < S_B$, compute

$D_o' = \alpha - \beta R_o$, when $R_o \geq 200$,

$D_o' = \alpha - \beta R_o + \delta(200-R_o)$, when $R_o < 200$,

where $\alpha = -0.125445 + 0.13625L + 0.01842L^2$;

$$\beta = \begin{cases} 0.001823 - 0.000403L \ , \ L \le 4 \ , \\ 0.000208 - 0.000128(L-4) \ , \ L > 4 \ ; \end{cases}$$

$$\delta = 1.5481 \cdot 10^{-3} - 5.068 \cdot 10^{-4} L + 4.923 \cdot 10^{-5} L^2 \ .$$

Now let

$$D_o = \begin{cases} D_o' \ , \ H_B \le 0.5 \ , \\ D_o' + D_H \cdot H_B \ , \ H_B > 0.5 \ , \end{cases}$$

where $D_H = 0.0389 - 0.0121L$.

And let

$$D_F = 10^{D_o} \ .$$

Finally, let

$$D_r = D_{asy} - D_F \ ,$$

and

$$D' = \frac{W \cdot 10^{D_r}}{R_o^2}$$

and

$$D = D' \cdot \frac{F}{0.5} \ ,$$

where F is the weapon fission fraction.

C. COMMENT

The underlying motivation of the above schema was obtained by the [illegible] that the dose as a function of R_o asymptotically [illegible] an expression of the form

$$D = \frac{A \cdot W \exp(-R_o)}{R_o^2} \ ,$$

which would be obtained from a point source with no fireball rise and constant absorbing cross section. The dose becomes close to

this asymptotic expression at dose levels of 100 to 1,000R. Thus the first effort is to obtain a linear fit at far ranges for

$$\log_{10} D_{asy} = \log_{10}\left[\frac{DR_o^{\,2}}{W}\right].$$

This was done assuming the same slope for all asymptotic curves. The fit was first made with $H_B = 0$; a correction for height of burst was then added. The height of burst correction ranged from about 30 percent (at maximum height of burst) for 0.1 KT yields to somewhat over 2 at large yields.

Using the "asymptotic dose," the logarithm of the ratio of asymptotic to actual dose, D_F, was estimated. This is a function that has high values for low slant ranges and decreases to zero as the two doses approach each other. D_F as a function of $1/R_o$ is almost linear near the origin, followed by a segment which, for most yields, was approximated by a parabola that is tangent to the linear piece at their intersection. The intersection occurs where the slant range has a value, SRB, that was determined by inspection from graphs of the function. At a particular height of burst, the linear segments for all yields below 250 KT could be taken, without too much forcing, to have one common intersection, for larger yields to have another different common intersection. This naturally separated the calculations into two ranges of yields below 250 KT and above 250 KT. These intersections were height of burst dependent for low yields, but could be taken as constant for high yields. (The ordinate of the intersection is negative, which results from errors in estimating the asymptotes. In effect, the estimation of D_F also partially compensates for errors in the asymptote estimate, and gives a two-step correction.)

The slope of the linear sections was represented by a linear function of height of burst for low yields, with the coefficients for the linear function yield dependent. For the high yields, no height of burst sensitivity was needed.

For small ranges, large values of $1/R_o$, a parabolic segment was added to the linear variation whose coefficient was yield

dependent in the low yield range. For the large yield range, this procedure gave an inadequate fit, so for low values of R_0 an alternative procedure was used, namely estimating $\log_{10}(D_r)$ as a function of R_0. A linear function was adequate, except for values of D_r under 200 feet, where a parabolic segment was added.

The algorithm used may seem a rather jerry-built assemblage of curve fitting procedures, as in one sense it is. The numerical values were obtained either from graph paper or simple least squares polynomial fits. The rationale for this approach is that a function of three variables is to be fit, and there is no a priori way of determining the functional forms needed for efficient fitting. The variation of dose as a function of slant range was, in fact, well approximated as a ratio of two polynomials. Unfortunately the coefficients of these polynomials did not systematically vary as a function of yield, or height of burst, rendering the development of an approximation valid for any yield or height of burst difficult. A simultaneous estimation technique with all three independent variables included seemed required. Although this was not attempted, it appeared likely that rather high order terms would be needed for any adequate polynomial approximation. Thus the method of "cut and fit" seemed more appropriate.

The original algorithm and the approximation were implemented on a Control Data 6400 computer, and compared over a range of yields, heights of burst, and slant ranges. The average time per calculation of all three types of doses for the original algorithm was 0.640 seconds, and for the approximation 0.00176 seconds.

A display of the accuracy of the approximation is presented in Table 1 where the minimum and maximum values of the ratio of fission product of doses computed by the approximation to that computed by the numerical integration is presented for various yields and scaled heights of burst over slant ranges of interest. The slant range of interest for this table is defined as any slant range where the fission product gamma dose is over 10R, and where 20 times the maximum of either the estimated or actual fission product gamma dose is less than the neutron dose. As can be seen,

Table 1. MINIMUM AND MAXIMUM RATIOS OF ESTIMATED TARGET DOSES

Yield (KT)	Scaled Height of Burst (ft/(KT)$^{1/3}$)					
	0	100	180	400	750	1250
0.1	0.99 0.99	0.99 0.99	0.99 0.99	1.00 1.00	1.00 1.00	1.00 1.00
1	0.98 0.99	0.98 1.00	0.99 1.00	0.99 1.00	0.99 1.00	0.99 1.00
10	0.98 1.03	0.98 1.03	0.98 1.02	0.98 1.01	0.97 0.99	0.94 0.96
40	0.87 1.11	0.90 1.10	0.92 1.09	0.95 1.06	0.95 1.02	0.98 1.02
100	0.92 1.17	0.96 1.08	0.98 1.07	0.99 1.04	0.99 1.02	1.02 1.03
300	0.97 1.05	0.97 1.04	0.97 1.04	0.97 1.03	0.95 1.02	-- --
1,000	0.94 1.11	0.95 1.07	0.95 1.08	0.89 1.12	0.94 1.10	-- --
10,000	0.50 0.99	0.50 1.12	0.49 1.10	0.51 1.10	0.79 1.30	-- --
30,000	1.06 0.66	1.57 0.71	0.76 1.24	0.69 0.95	-- --	-- --

the difference is generally within 10 percent of the fission product gamma dose except for the yields of 10MT and 30MT. For these larger weapons, however, the overpressures at the dose ranges of interest are generally well over 30 psi. As can be seen in Table 1, and as is even more evident from listings as a function of slant range, the errors are quite systematic. Thus, if desired, further corrections could be readily developed to make the estimated error still closer to the actual error. Such corrections would require possibly a 20 percent to 50 percent increase in calculation time for each subroutine call. Use of this multiple approximation technique is not untypical of this approach, where the error bounds achieved are often dependent primarily on the effort expended in developing the approximations.

```
C
C     CALCULATE PROMPT INITIAL NUCLEAR RADS FROM BOMBS
C     [illegible]
C     CALLED [illegible]
C
C     THIS IS A RAPID RUNNING VERSION USING CURVE FITTING TO THE
C        RESULTS OF THE FISSION PRODUCT GAMMA CALCULATION WHICH REQUIRES
C        A NUMERICAL INTEGRATION TO ACCOUNT FOR THE RISING FIREBALL
C
C     [illegible]
C     [illegible]
C     [illegible]
C
C     [illegible]
C     [illegible]
C        THE CALLING PROGRAM
C
      DIMENSION [illegible]
      [illegible]
      DATA [illegible]
      DATA [illegible]
      DATA [illegible]
      DATA [illegible]
      DATA [illegible]
      DATA [illegible]
      DATA [illegible]
C
C     [illegible]
C     [illegible]
C     [illegible]
C
C     [illegible]
C     [illegible]
C     [illegible]
C
C     [illegible]
C
C     [illegible]
```

PRECEDING PAGE BLANK-NOT FILMED

```
121   CONTINUE
C
C
C     START ESTIMATING ASYMPTOTE
C        ESTIMATE TO [illegible]
      [illegible]
     1 [illegible]
      [illegible]
      IF([illegible]) GO TO 111
      [illegible]
     1 [illegible]
      [illegible]
     1 [illegible]
      [illegible]
      [illegible]
111   CONTINUE
C
C
C     [illegible] ESTIMATE DIFFERENCE BETWEEN ASYMPTOTE AND REAL CURVE
      [illegible]
      IF([illegible]) GO TO 125
C     FOR LOW YIELDS [illegible] IS CLOSE TO STRAIGHT LINE
C        [illegible]
C        [illegible]
      [illegible]
C     SET A COMMON INTERCEPT FOR ALL LINES
      [illegible]
      [illegible]
C     SLOPE OF LINES TAKEN AS LINEAR FUNCTION OF [illegible]
C        WITH THIS FUNCTIONS SLOPE AND INTERCEPT YIELD DEPENDENT
      IF([illegible]) GO TO 122
      [illegible]
      [illegible]
      GO TO 123
122   CONTINUE
      [illegible]
      [illegible]
123   CONTINUE
      [illegible]
      IF([illegible]) GO TO 131
      [illegible]
      GO TO 129
131   CONTINUE
C     START PARABOLIC PART [illegible]
      IF([illegible]) GO TO 132
      [illegible]
      [illegible]
      [illegible]
      GO TO 133
132   CONTINUE
      [illegible]
      [illegible]
      [illegible]
133   CONTINUE
      [illegible]
      [illegible]
      [illegible]
      GO TO 129
129   CONTINUE
C     [illegible]
      IF([illegible]) GO TO 126
```

SUBROUTINE PDIST (NTYPE, LOGV, PRES, DIST)

A. GENERAL DESCRIPTION

This subroutine computes the distance (DIST) in nautical miles at which a given value of overpressure (PRES) in Psi occurs from a 1 MT detonation. The distances are based on a fit to the data in Glastons "The Effects of Nucler Weapons," published by the AEC in 1962. The data were taken from the Height of Burst curves on pages 137 and 139. Three Height of Burst options are available as determined by the parameter NTYPE. They are:

NTYPE = 1, Surface Burse

NTYPE = 0, Burst Height maximized distance for 10 psi contour, i.e., 7,400 ft

NTYPE = -1 The distance computed is the distance at the knee of the height of burst curve, i.e., for the height of burst which gives the largest distance at the calling pressure.

If the parameter LOGV is set equal to one, the input parameter PRES is interpreted as LOG_{10}(PRES), a value of 0 is interpreted as the regular pressure. This feature may, on occasion, allow saving taking of a logarithm.

The formulas for pressure as a function of distance are the same as in the subroutine PROMPT except that the inverse functions are calculated there.

B. REQUIREMENTS ON THE CALLING PROGRAM

If the subroutine is called with a value of PRES less than 0.00001, for LOGV=1, or PRES less than -5 for LOGV=1, a distance of 99999. is returned. If the subroutine is called for a 10 psi airburst (NTYPE=0) and a pressure greater than 38 psi, a distance of 0 is returned. In either case no error message is given. The only communication of this subroutine with the calling program is through the calling parameters.

C. ALGORITHM IMPLEMENTED

For a surface burst or 10 psi airburst, the equation for pressure as a function of distance is given by

$$\log_{10} \text{dist} = A + B \log_{10}(\text{PRES}/20)$$

For surface burst (NTYPE = 1)

A = .0596 (corresponds to a distance of 1.147 miles at 20 psi

B = -2.255 $p \geq 20$ psi

= -1.825 $p < 20$ psi

For Optimum Burst Height to maximize pressure (NTYPE= -1)

A = .1903 (corresponds to a distance of 1.55 miles at 20 psi

B = -1.8303 $p \geq 20$ psi

= -1.52 $p < 20$ psi

For 10 psi Optimized Burst Height (NTYPE=0)

Dist = 0	38 psi $\leq$ PRLS
Dist = (38-PRES)19.5	20 psi $\leq$ PRES $\leq$ 38 psi
Dist = 0.922 +(20-PRES)/5.1	15 psi $\leq$ PRES $\leq$ 20 psi
$\text{Log}_{10}(\text{Dist}) = 0.2787 + (\log_{10}(15/\text{PRES}))1.75$	PRES < 15 psi

(This corresponds to a distance of 1.9 miles at a pressure of 15 psi.)

These equations fit the pressure to within 10 percent over the range of 1 to 200 psi.

```
      SUBROUTINE PDIST(NTYPE,LOGV,PRES,DIST)
C     NEVUMS STANDARD
C     LAST REVISION OCT. 30, 1972
C     COMPUTES DISTANCE GIVEN PRESSURE
C     DISTANCE  IS IN NAUTICAL MILES, PRESSURE IS PSI. ONE MT. WEAPON
C     YIELD IS ASSUMED. BASED ON A FIT TO HEIGHT OF BURST CURVES IN
C     EFFECTS OF NUCLEAR WEAPONS. THE SAME EQUATIONS ARE USED FOR THE
C     INVERSE CALCULATION IN THE SUBROUTINE PROMPT.
C     FIT TO WITHIN 10 PER CENT FROM 1 TO 200 PSI
C     NTYPE = 1 IS SURFACE, EQUAL 0 IS 10PSI OPTIMIZED,=-1 IS OPT
C     AIRBURST
C     IF LOGV = 0 CALL WITH PRESSURE. IF 1 CALL WITH LOG PRESS

      IF (LOGV .EQ. 1) GO TO 8
      IF (PRES .GT. 0.000001) GO TO 9
      DIST = 99999.
      RETURN
    8 CONTINUE
      IF( PRES .GT. -6.) GO TO 9
      DIST = 99999.
      RETURN
    9 CONTINUE

      IF( NTYPE .LE.0) GO TO 20

C     SURFACE BURST
      IF( LOGV .EQ. 1) GO TO 11
      ALPRES = ALOG10(PRES)
      GO TO 12
   11 CONTINUE
      ALPRES = PRES
   12 CONTINUE
      IF( ALPRES .LE. 1.30103) GO TO 13
      SLOPE = -2.265
      GO TO 14
   13 CONTINUE
      SLOPE = -1.495
   14 CONTINUE
      ALDIST = 0.3030 + ( ALPRES - 1.30103)/SLOPE
      DIST = 10.**ALDIST
      RETURN
   20 CONTINUE
      IF( NTYPE .LT. 0) GO TO 50

C     10 PSI OPT AIRBURST
      PPE = PRES
      IF( LOGV .EQ.1) GO TO 21
      IF( PRES .GT. 15.)    GO TO 30
      ALPRES = ALOG10(PRES)
      GO TO 40
   21 CONTINUE
```

```
      ALPRES = PRPS
      IF( ALPRES .LT. 1.1761) GO TO 40
      PRE  = 10.**ALPRES
      GO TO 30
   30 CONTINUE
C     NO DAMAGE EXIT IF PRESSURE GREATER THAN 30PSI
      IF( PRE  .LE. 30.) GO TO 31
      DIST = 0.
      RETURN
   31 CONTINUE
      IF( PRE .LT.20.) GO TO 32
      DIST = ( 30. - PRE )/10.5
      RETURN
   32 CONTINUE
      DIST = 0.952 + ( 20. - PRE )/8.1
      RETURN

   40 CONTINUE
C     LOW PRESSURE
      ALDIST = .2707 + (1.1761- ALPRES)/1.75
      DIST = 10.**ALDIST
      RETURN

C     OPT AIRBURST
   50 CONTINUE
      IF( LOPT .EQ. 1)  GO TO 51
      ALPRES = ALOG10(PRES)
      GO TO 55
   51 CONTINUE
      ALPRES = PRPS
   55 CONTINUE
      IF( ALPRES .GT. 1.30103) GO TO 52
      SLOPE = -1.59
      GO TO 53
   52 CONTINUE
      SLOPE = -1.490
   53 CONTINUE
      ALDIST = .1903 + ( ALPRES - 1.30103)/SLOPE
      DIST = 10.**ALDIST
      RETURN
      END
```

SUBROUTINE PROMPT (DECK #49)

A. GENERAL DESCRIPTION

This subroutine computes values of overpressure, initial nuclear radiation, and initial thermal energy, given values of weapon yield and distance. Communication with the main program is through the block common [redacted] to possibly shorten the calling time since this subroutine may be repeatedly used. The parameter JHTPR (called NTYPE internally) selects height of burst options.

These are

JHTPR = 1 Surface burst

JHTPR = 0 Height of burst to maximize the distance on the ground for the 10 psi contours. This height of burst is 7400 ft for a 1 MT weapon

JHTPR = -1 The height of burst is varied for every distance so that at that distance the height of burst maximizes the pressure obtained. The pressures are always those at the area of the height of burst curves. This option is implemented for the pressure calculation only. ~~For calculation of initial nuclear radiation and thermal radiation the NTYPE=0 option is assumed~~.

If the parameter JTINR (called LONG internally) is 0 only the pressure calculation is carried. If JTINR is equal to 1 the normal initial nuclear radiation and thermal radiation calculations are carried out. In these initial nuclear radiation calculations an ambient air density of 1.11 is assumed. If JTINR equals 2, a special initial nuclear radiation calculation is performed which a file of data made for one of 3 yields, 10 KT, 300 KT, or 1 MT and for three air densities, 1.1, 1.2, or 1.3.

B. REQUIREMENTS ON THE CALLING PROGRAM

Since this routine may be used rather often, the calling program is required to monitor the communications through the block common EFFCAL. The parameters JHTPR and JTINR must be defined for each call for addition. The following information must be supplied or is returned:

For JTINR = 0

Supply	YLDPCR	cube root of the yield
"	DSTP	distance in nautical miles
Obtain	PRESP	pressure in psi
"	[redacted]	logarithm to the base 10 of the pressure.

For JTINR = 1

The above plus

Supply	YLDP	weapon yield MT
"	THRPVS	thermal visibility in units of [1/meters]
Obtain	RADIP	initial nuclear radiation (rads)
"	THERP	thermal radiation (cal/cm^2)

For JTINR = 2

Supply The above plus DENIP (ambient air density). IN THIS CASE

The yield must have a value of .04, .3 or 1 MT; if none of these, a default value of .04 is assumed.

The air density must be 1.1, 1.2 or 1.3; if none of these, a default value of 1.3 is assumed.

C. ALGORITHMS IMPLEMENTED

For overpressure the pressure distance relations are obtained from a prt of the data in the Height of Burst curves in "Effects of Nuclear Weapons," 1962. The equations are the inverse functions

of those implemented in the subroutine PDIST. The input distance DSTP is divided by the cube root if the yield   obtain the 1 MT distance DIST. To obtain the pressure in psi given the distance in nautical miles for surface bursts (NTYPE = 1)

$$\log_{10}(\text{PRES}) = A + B \log_{10}(\text{DIST}/1.147)$$

where

$A = 1.301$ (equivalent to 20 psi)

$$B = \begin{cases} -2.255 & \text{DIST} \leq 1.147 \\ -1.825 & \text{DIST} > 1.147 \end{cases}$$

For maximum air burst distance (NTYPE = -1)

$$\log_{10}(\text{PRES}) = A + B \log_{10}(\text{DIST}/1.55)$$

where

$A = 1.301$ (equivalent to 20 psi

$$B = \begin{cases} -1.830 & \text{DIST} \leq 1.55 \\ -1.52 & \text{DIST} > 1.55 \end{cases}$$

For 10 psi optimized air burst (NTYPE = 0)

$$\text{PRES} = 38 - 19.5 \cdot \text{DIST} \qquad 0 \leq \text{DIST} < .922$$

$$\text{PRES} = 20 - 5.1(\text{DIST} - .922) \qquad .922 \leq \text{DIST} < 1.90$$

$$\log_{10}(\text{PRES}) = 1.1761 - 1.75 \log_{10}(\text{DIST}/1.90) \qquad 1.90 \leq \text{DIST} \leq \infty$$

The pressure is forced to ~~logarithm~~ stay within the limits of .001 to 99999 psi.

The error in the fit is within 10 ~~psi~~ % over the range from 1 to 200 psi.

The general calculation for initial nuclear radiation is based on a fit of radiation to slant range of the form

$$RADP = ARD \exp(-BRD \cdot SR)/SR^2$$

where

SR is the slant range in meters

$ARD = Yield \cdot (C - 3.2 \times 10^{12} \log_{10}(yield)$

with

$C = 3.2 \times 10^{12}$ JFITRR = 1

$C = 7.25 \times 10^{12}$ JFTPR = 0 or -1

$BRD = C_B - 0.82 \times 10^{-3} \log_{10}(yield)$

with

$C_B = 3.35 \times 10^{-3}$ JFTPR = 1

$C_B = 3.02 \times 10^{-3}$ JFTPR = 0 or -1

The special calculation has a fit of the form

$\log_{10} rad = A + B \log_{10}(SR)$

The values of A+B for the 9 different combinations of yield in air density are available from the coding sheets.

The maximum allowed value of radiation 10^8 rads.

The thermal radiation is calculated from the equation

$THER = 1/3\ Y \exp(-THRPVS \cdot SR)/(4\pi \cdot SR^2)$.

The thermal radiation is limited to 10^8 cal/cm^2 .

```
      SUBROUTINE [illegible]
C     [illegible] STANDARD
C     LAST REVISED OCT. 31, 1972
C      TO CALCULATE PRESSURE(PSI), THERMAL RADIATION(CAL/CM2),
C      AND INITIAL NUCLEAR RADIATION(RAD) GIVEN DISTANCE(IN MI.)
C   NTYPE = 1 SURFACE BURST, 2 IS 15PSI OPTIMIZED AIRBURST
C     IF -1 IS A OPTIMUM AIRBURST HEIGHT
C     LONG = 1 DO WHOLE CALCULATION, 0 DO PRESSURE ONLY
C     LONG = 2 DO SPECIALIZED IND CALCULATION.
      COMMON/[illegible]/ [illegible]
     1 [illegible]
      DATA [illegible]
      DATA THIRD/ 0.3333333/
      DIST = DSTP/YLDRCB
      IF( DIST .GT. 0.00001) GO TO 4
      DIST = 0.00001
    4 CONTINUE
      IF ( NTYPE .LE. 0) GO TO 40
C     SURFACE BURST
      ALDIST = ALOG10(DIST)
      IF( DIST .GE. [illegible]) GO TO 2
    1  SLOPE = -2.[illegible]
       GO TO 3
    2 CONTINUE
      SLOPE = - 1.[illegible]
    3  CONTINUE
       ALPRES = 1.3010 + SLOPE*(ALDIST - [illegible])
       PRES = 10.** ALPRES
      GO TO 60
   40 CONTINUE
      IF( NTYPE .LT. 0) GO TO [illegible]
C     15 PSI OPTIMIZED BURST
      IF( DIST .LT. [illegible]) GO TO 41
      ALDIST = ALOG10(DIST)
      ALPRES = [illegible] - [illegible]*(ALDIST - [illegible])
      PRES = 10.**ALPRES
      GO TO 60
   41 CONTINUE
      IF( DIST.LT. [illegible]) GO TO 42
      PRES = [illegible] - [illegible]*(DIST - [illegible])
      ALPRES = ALOG10(PRES)
      GO TO 60
   42 CONTINUE
      PRES = [illegible] - [illegible]*DIST
      ALPRES = ALOG10( PRES)
      GO TO 60
   43 CONTINUE
```

```
C     OPTIMIZED AIRBURST HEIGHT
      ALDIST = ALOG10(DIST)
      IF(DIST .LT. 1.55) GO TO 92
      SLOPE = -1.52
      GO TO 91
 92   CONTINUE
      SLOPE = - 1.490
 91   CONTINUE
      ALPRES =  1.3910 - SLOPE*(ALDIST -  .1903)
      PRES =  10.**ALPRES
 74   CONTINUE
       IF( PRES .LT. PMIN) PRES = PMIN
       IF( PRES .LT. PMIN) ALPRES = ALPMIN
       IF(PRES .GT. PMAX) PRES = PMAX
       IF( PRES .GT. PMAX) ALPRES = ALPMAX
      IF( LONG .NE. 0) GO TO 72
      RETURN

C     NO. 70 [illegible]
 72   CONTINUE
      IF( LONG .EQ.2) GO TO 80
C     NORMAL IOH CALCULATION
      IF( NTYPE .EQ. 1) GO TO 70
C     10 PSI OR OPT AIR BURST
      CMD = 7.25E+12
      CMUH = 3.82E+3
      SHS = 5.87E+6 - 3.02E+6*DIST*DIST
      SH = SQRT(SHS)
      GO TO 71
 70   CONTINUE
C     SURFACE BURST
      CMD = 3.2E+12
      CMUH = 3.39E+3
      SH = 1490.*DIST
      SHS = SH*SH
 71   CONTINUE
      SHD = YIELD*(CMD - 3.2E+12*LUPLS)
      SHH = CMUH - 1.8E+3*LUPLS
      SHD = SHD*[illegible] /SHS
      IF( SHD .GT. 1.E+9) SHD = 1.E+9
      GO TO 11
 80   CONTINUE
C     SPECIALIZED IOH CALCULATION
C     TO OBTAIN IOH DIRECTLY FROM SLANT RANGE
C     SH IS VALUE OF SHS IN THIS CALCULATION
      IF (NTYPE.EQ.2) GO TO 81
C     SURFACE BURST
C     THIS CANNOT HANDLE AIR BURSTS OTHER THAN 10 PSI OPT
      SLNTR = DIST*DIST + .0020114
      SLNTR = SQRT(SLNTR)
      GO TO 82
```

```
      41  CONTINUE
      C   IS WSI 80?
          IF( YIELD .EQ. 1.) GO TO 43
          IF (YIELD .EQ. .3)GO TO 44
      C   40 KT
          SLNTR = DIST*DIST + 0.170416
          SLNTR = SQRT(SLNTR)
          GO TO 42
      43  CONTINUE
          SLNTR = DIST*DIST + 1.0412
          SLNTR = SQRT(SLNTR)
          GO TO 42
      44  CONTINUE
          SLNTR = DIST*DIST + 3.99677
          SLNTR = SQRT(SLNTR)
      42  CONTINUE
          IF( YIELD .EQ. .04) GO TO 61
          IF( YIELD .EQ. .3) GO TO 62
      C   YIELD IS 1MT NOW
          IF( SW .EQ. 1.1) GO TO 63
          IF( SW.EQ. 1.3) GO TO 64
          SLOPE = -4.0733
          SHAFT = 8.6[illegible]
          GO TO 69
      63  SLOPE = -3.6634
          SHAFT =  1.0197
          GO TO 69
      64  SLOPE = -3.7021
          SHAFT = .3047
          GO TO 69
      C   YIELD = .3
      62  SLOPE = -4.9362
          SHAFT =  9.7776
      69  CONTINUE
          SLANSD = 10. ** SLOPE*(SLNTR - SHAFT)
          GO TO 184
      61  CONTINUE
      C   YIELD = 40 KT
          IF( SW.EQ. 1.1) GO TO 101
          IF(SW .EQ. 1.3) GO TO 102
      C   SW = 1.3
          IF(SLNTR .GT.0.26139) GO TO 103
          SLOPE = -21.1376
          SHAFT = 1.26170
          YALLA = 5.10613
          GO TO 119
      103 CONTINUE
          SLOPE = -4.0294
          SHAFT = .0663
          YALLA = 0.
          GO TO 119
      102 CONTINUE
          IF( SLNTR .GT. 0.26000) GO TO 104
          SLOPE = -17.8663
          SHAFT = 0.27664
```

```
          VALLS = 9.1E+13
          GO TO 110
    105   CONTINUE
          SLOPE = -9.5761
          SAVAL = .5672
          VALLS = 0.
          GO TO 110
    102   CONTINUE
          IF(SLNTH .GT. 0.31290) GO TO 104
          SLOPE = -10.0493
          SAVAL = 0.3129
          VALLS = 9.1E+13
          GO TO 110
    104   CONTINUE
          SLOPE = -6.7496
          SAVAL = 0.5900
          VALLS = 0.
    110   CONTINUE
          SLOWAD = VALLS + SLOPE*(SLNTH - SAVAL)
    120   CONTINUE
          IF( SLOWAD .GT. 4.) GO TO 121
          WAD =  10.**SLOWAD
          GO TO 11
    121   WAD = 1.E+4
    11     CONTINUE

C          NOW THERMAL RADIATION
          THRM = [illegible]
          IF( THRM .GT. 1.E+4) THRM = 1.E+4
          RETURN
           END
```

SUBROUTINE RADKL (DECK #80)

A. GENERAL

The subroutine was originally developed for the program ANDANTE to summarize the effects of Blast and Initial Nuclear Radiation on a single city. Due to its specialized nature, and since it only produces output, it is not fully documented here. The subroutine obtains for each of a list of weapons the distance to each tract of a city and then computes probability of blast kill and injury directly from kill and injury arrays. The subroutine PROMPT furnises values of initial nuclear radiation which are then converted into probability of radiation kill. Finally, a variety of formats are used to exhibit the compiled results.

B. REQUIREMENTS ON CALLING PROGRAM

The calling program must supply the following block common variables. Definitions of the variables are contained in the block common descriptions.

/TMPAND/	Only LSTAPE is needed. If this control parameter is 1 the output is also sent to an output tape.
/WPNPRB/	NWP - number of weapons XZ - X weapon coordinate in naut. mile YZ - Y weapon coordinate in naut. mile
/ST44TA/	X,Y coords of tracts and POP of each track in naut. miles and same coordinate system as XZ, YZ
/CITYPR/	NAMEC(20) - city name NTACTS - number of tracts
/VULPR/	D50 mean lethal radiation dose SIG50 standard deviation of SIG50 C50 mean injury radiation dose

	SIGLS0	standard deviation of SIG50
	PF	entered as 1st element of PFILV; initial nuclear radiation protection factor
/WPNPR/	REL	weapon reliability
	[illegible]	cube root of weapon yield
/PKPR/	kill probability and injury probability tables from FLPKHU or FLPKA	
/IOPR/	MT	tape assignment if LSTAPE = 1

The subroutine uses /OPTLAV/ to communicate with subroutine PROMPT. The subroutine PROMPT (and CUMNOR) must be available.

C. ALGORITHMS IMPLEMENTED

For each weapon the probability of blast kill is computed as in the subroutine ONEPAS, i.e., by linear interpolation if the PK-distances squared array. The probability of blast injury is computed the same except that a probability of injury distance squared table is used. From several weapons the survival or noninjury probability is the product of [illegible] probabilities for all weapons.

For nuclear radiation the probability of radiation kill or injury is found by accumulating radiation doses for all weapons to the present one and using the cumulative normal function with appropriate values of mean lethal or mean injury dose, and standard deviations, to find the total kill.

These calculations are done between lines 88 and 125 in the program.

Combined effects are computed by assuming in most cases the probabilities of the various effects are independent,with the obvious condition such as no killed person can be listed as injured

imposed. In some calculation attempts are made to include synergistic effects of radiation and blast by assuming if a person is injured by both blast and nuclear radiation he is killed. These calculations are made from line 125 to 160. The remainder of the subroutine output results in various formats which hopefully are self explanatory.

```
      SUBROUTINE RADAL

C     NEVUNS COMPATABLE
C     LAST REVISED NOV. 16, 1972

C     COMPUTES AVERAGE KILL BY RADIATION
C     LETHAL AGENTS ARE BLAST AND THERMAL RADIATION

      DIMENSION RADAR(1000)  ,RELRAD(1000)
      DIMENSION RTAB(13,100)
      DIMENSION [illegible](4,100), [illegible](5) , [illegible](5)
      COMMON/[illegible]/[illegible],NAME,[illegible],[illegible],[illegible],ADJ[illegible],LSTAGE,LSTC
     1 ,[illegible]
      COMMON/[illegible]/[illegible](150),[illegible](150),[illegible](300)
      COMMON/[illegible]/[illegible](6000),[illegible](6000),POP(6000),[illegible](6000)
      COMMON/CITPOP/NAME(2),[illegible](2), TOTPOP,[illegible](30),[illegible]
      COMMON/[illegible]/ [illegible](8),[illegible],SIG[illegible],[illegible],SIG[illegible], [illegible](4),[illegible],
     1 [illegible](10), [illegible](30)
      COMMON/[illegible]/ [illegible](2),[illegible],[illegible](7), NTYPA,[illegible](3),YLDHA,[illegible]
      COMMON/[illegible]/[illegible], [illegible](30),[illegible](30),[illegible](30),[illegible](30),
     1 [illegible](30),[illegible](30),[illegible](30),[illegible](30) ,[illegible]
      COMMON/[illegible]/[illegible],YLDHU,[illegible],LOWG,NTYPE,DIST,[illegible],PSI,[illegible],THRM,
     1THRMVS, [illegible], [illegible](3)
      COMMON/[illegible]/[illegible],[illegible],[illegible](7),[illegible],[illegible](8)


      YLDHU = YLDHA
      NTYPE = NTYPA
C     ASSUME INFINITE VISIBILITY TO GET MAX THERMAL DOSEAGE
      THRMVS = 1.
      LOWG = 1
      PSYHOB = 1.
      PCGAL = .5
      WRITE(MO,3)
    3 FORMAT(1H1,//,50X, 18HTOTAL CITY EFFECTS    )
      WRITE(MO,4)(NAME(I),I = 1,2)
    4 FORMAT(1H0, 20X, 13HCITY NAME IS    ,2A10)
      WRITE(MO,35)
   35 FORMAT(1H0)
      WRITE(MO,30)
   30 FORMAT( 1H ,[illegible],10HFATALITIES  [illegible],[illegible], 8HINJURIES
     1 [illegible], 10H  COMB FATALITIES   ,[illegible] SYNERGISTIC FATL       )
      WRITE(MO,31)
   31 FORMAT( 1H , [illegible],[illegible]POPULATION ,[illegible], [illegible], [illegible]BLAST, [illegible] ,
     1              [illegible],[illegible]POPULATION ,[illegible], [illegible], [illegible]BLAST, [illegible] ,
     2    ,[illegible] RAD. AND BLAST    ,[illegible] RAD. AND BLAST          )
      WRITE(MO,32)
   32 FORMAT( 1H   ,[illegible], [illegible] LST [illegible]   TOTAL        ,[illegible]   TOTAL   LST [illegible]
     1              ,[illegible], [illegible] LST [illegible]   TOTAL        ,[illegible]   TOTAL   LST [illegible]
     2      ,[illegible]     ,    [illegible] LST [illegible]   TOTAL        ,[illegible] LST [illegible]   TOTAL  )
      IF (LSTAGE .NE.1) GO TO 40
      WRITE(MT,3)
      WRITE(MT,35)
      WRITE(MT,30)
      WRITE(MT,31)
```


PRECEDING PAGE BLANK-NOT FILMED

```
      WRITE(NT,32)
   99 CONTINUE
      DO 5 I = 1,NTRCTS
      V(I) = POP(I)
      [illegible](I) = POP(I)
    5 [illegible](I) = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] =0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      DO 10 II = 1,[illegible]
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] =0.
      [illegible]     = 0.
      [illegible] = 0.
      [illegible] =     0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] =     0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      [illegible] = 0.
      DO 17 JJ = 1,NTRCTS
      T[illegible] = X[illegible](II) - X(JJ)
      T[illegible] = Y[illegible](II) - Y(JJ)
      DISTS = [illegible] + [illegible]
      DIST = SQRT(DISTS)
      CALL [illegible]
      [illegible](JJ) = [illegible](JJ) + [illegible]
      TT = [illegible](JJ)[illegible]
      TTT = (TT - [illegible])/[illegible]
      [illegible] [illegible](TTT)
      [illegible] = POP(JJ) * [illegible]
      TTI = ( TT - C[illegible])/[illegible]
      [illegible] = [illegible](TTT)
      [illegible] = POP(JJ)*[illegible] + [illegible]
      [illegible] = [illegible] - [illegible]
      [illegible] = 1. - [illegible]
      [illegible] = [illegible]
C     TO SPEED UP CALC - NEEDS [illegible](2) = 0.
      IF( [illegible](2) .[illegible].00) GO TO 102
      [illegible] = 0.
      GO TO 103
  102 CONTINUE
      J = 0
  1[illegible] CONTINUE
```

```
      J = J + 1
      IF(IORDR(J).GT. 00) GO TO 1[illegible]
      P   = PACT(J) + 100 - DSDR(J)*DELPSR(J)/DELDSR(J)
      P[illegible] = PODEL
 103  CONTINUE
      IF( [illegible](2) .GT. 00) GO TO 100
      P[illegible] = 0.
      GO TO 1[illegible]
 100  CONTINUE
      J = 2
 1[illegible]  CONTINUE
      J = J + 1
      IF(IORDR(J).GT. 00) GO TO 100
      P   = PACT(J) + 100 - DSDI(J)*DELPSI(J)/DELDSI(J)
      P[illegible] = PODEL
 1[illegible]  CONTINUE
      [illegible] = [illegible] + V(JJ)*P[illegible]
      [illegible] = [illegible] + DELDA(JJ)*P[illegible]
C     REDUCE  NUMBER OF BLAST SURVIVORS AND UNINJURED
      V(JJ) = V(JJ) *(1. - P[illegible])
      DELDA(JJ) = DELDA(JJ)*(1. - P[illegible])
      [illegible] = [illegible] + V(JJ)*P[illegible]
      [illegible] =  V(JJ) - DELDA(JJ)
C     ASSUME THAT IF INJURED BY BOTH BLAST AND RADIATION THIS IS FATAL
C     FOR SYNERGISTIC EFFECTS CALCULATION. OTHER DATA CAN BE USED HERE
C     IF DESIRED
      [illegible] = [illegible] + [illegible]  P[illegible]     [illegible]
      [illegible] = POP(JJ) - V(JJ)
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] =  [illegible] + [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] =  [illegible] + [illegible]
      [illegible] = [illegible] + DELDA(JJ) [illegible]
      [illegible] =  [illegible] + DELDA(JJ)*[illegible]
      [illegible] = [illegible] + DELDA(JJ) [illegible]
 [illegible]  CONTINUE
      [illegible] = [illegible] - [illegible]
      [illegible] = [illegible] - [illegible]
      [illegible] = [illegible] - [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] - [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] - [illegible]
      [illegible] = [illegible] + [illegible]
      [illegible] = [illegible] - [illegible]
      IF([illegible].LT.[illegible]) GO TO 1[illegible]
      [illegible] = [illegible] + ([illegible])*(1.- [illegible])
      [illegible] = [illegible] + [illegible]
 1[illegible]  CONTINUE
      WRITE([illegible])[illegible]
     1[illegible]
 [illegible]  FORMAT( [illegible] )
```

```
      IF (LSTAGE .NE.1) GO TO 61
      WRITE([illegible])
     1[illegible]
   61 CONTINUE
      STAB(1,III) = [illegible]
      STAB(2,III) = [illegible] - [illegible]
      STAB(3,III) = TOTPOP - [illegible]
      STAB(4,III) = [illegible]
      STAB(5,III) = [illegible] - [illegible]
      STAB(6,III) = [illegible]
      STAB(7,III)= 100.*STAB(6,III)/TOTPOP
      STAB(8,III) = TOTPOP - [illegible]
      STAB(9,III)= 100.*STAB(8,III) /TOTPOP
      STAB( 10,III) = TOTPOP - [illegible]
      STAB(11,III) = 100.*STAB(10,III) /TOTPOP
      STAB(12,III) = TOTPOP - [illegible]
      STAB(13,III) = 100.*STAB(12,III)/TOTPOP
      [illegible]( 1,III)= [illegible]
      [illegible]( 2,III)= [illegible]
      [illegible]( 3,III)= TOTPOP - [illegible] - [illegible]
      [illegible]( 4,III)= [illegible]
      [illegible]( 5,III)= [illegible]
      [illegible]( 6,III)= TOTPOP - [illegible] - [illegible]
      [illegible]( 7,III)= [illegible]
      [illegible]( 8,III)= [illegible]
      [illegible]( 9,III)= [illegible]
      [illegible](10,III)= [illegible]
      [illegible](11,III)= [illegible]
      [illegible](12,III)= [illegible]
      [illegible](13,III)= [illegible]
      [illegible](14,III)= [illegible]
      [illegible](15,III)= [illegible]
      [illegible](16,III) = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
      [illegible] = [illegible]
   10 CONTINUE
      WRITE([illegible])
   92 FORMAT(1H1,//,[illegible] TOTAL CITY EFFECTS )
      WRITE([illegible])
   94 FORMAT([illegible]CASUALTIES ,[illegible]INJURIES .[illegible]FATALITIES
     1 ,[illegible] ,
     2 [illegible] * * * * ,[illegible] ,
     3 [illegible] * * * * )
      WRITE([illegible])
   96 FORMAT( [illegible]
     1 [illegible]
     2 [illegible] )
      WRITE([illegible])
   98 FORMAT( [illegible]
     1 [illegible] ,[illegible]TOTAL, [illegible]PERCENT ,
     2 [illegible]TOTAL ,[illegible]PERCENT ,[illegible]TOTAL, [illegible]PERCENT,[illegible]
     3 [illegible]TOTAL, [illegible]PERCENT )
      IF (LSTAGE .NE.1) GO TO 97
```

```
            WRITE(NT, 8[illegible])
            WRITE(NT, 8[illegible])
            WRITE(NT, 8[illegible])
            WRITE(NT, 8[illegible])
      92    CONTINUE
            DO 81 IJ = 1,100
            WRITE(NO, 82) IJ,PTAB(1,IJ), PTAB(2,IJ) , PTAB(3,IJ), PTAB(4,IJ),
           1PTAB(5,IJ) ,PTAB(6,IJ) , PTAB(7,IJ), PTAB(8,IJ),  PTAB(9,IJ),
           2PTAB(10,IJ) , PTAB(11,IJ) , PTAB(12,IJ), PTAB(13,IJ)
      82    FORMAT( 1H  ,I3, [illegible]
           1F7.3   ,[illegible] F6.0,F7.3, [illegible] F6.0,F7.3   )
            IF (LSTAPE .NE.1) GO TO 83
            WRITE(NT, 82) IJ,PTAB(1,IJ), PTAB(2,IJ) , PTAB(3,IJ), PTAB(4,IJ),
           1PTAB(5,IJ) ,PTAB(6,IJ) , PTAB(7,IJ), PTAB(8,IJ),  PTAB(9,IJ),
           2PTAB(10,IJ) , PTAB(11,IJ) , PTAB(12,IJ), PTAB(13,IJ)
      83    CONTINUE
      81    CONTINUE
            WRITE( NO,70)
      70    FORMAT( 1H1, ///, 50X, 29HSTILL HOME TOTAL CITY EFFECTS      )
            WRITE(NO, 75)
      75    FORMAT(1H  ,30X,39HSINGLE EFFECTS  (A B)  ARE PERCENT AND 1 - PERC
           1ENT        )
            WRITE (NO,76)
      76    FORMAT( 1H0 , 11X, 1H=  [illegible]
           1,  1H=  [illegible]  ,10X, 1H=  [illegible]
           21H=  [illegible]  ,10X,10HCORRELATED     )
            WRITE (NO,73)
      73    FORMAT( 1H  , [illegible] 10HFATALITIES  ,[illegible] 9HINJURIES, [illegible]
           1[illegible]INJURED ,11X,  10HFATALITIES, [illegible]INJURIES   ,[illegible]
           2 [illegible]INJURED, 10X, 10HFATALITIES     )
            IF (LSTAPE .NE.1) GO TO 90
            WRITE(NT, 70)
            WRITE(NT, 75)
            WRITE(NT, 76)
            WRITE(NT, 73)
      90    CONTINUE
            DO 71 [illegible] = 1,100
            WRITE(NO,72) [illegible]
      72    FORMAT(1H0 ,[illegible]
            IF (LSTAPE .NE.1) GO TO 99
            WRITE(NT,72) [illegible]
      99    CONTINUE
            DO 77 I = 1,4
            EA(I) = 100.*[illegible]/TOTPOP
            EA(I[illegible]) = 100. - EA(I)
      77    CONTINUE
            WRITE(NO,74)(EA(I),EA(I[illegible]),I = 1,4)
      74    FORMAT( 1H  [illegible]
           1 [illegible]    )
            IF (LSTAPE .NE.1) GO TO 96
            WRITE(NT,74)(EA(I),EA(I[illegible]),I = 1,4)
      96    CONTINUE
      71    CONTINUE
            WRITE(NO,89)
      89    FORMAT(1H1, ///, 50X,[illegible] TOTAL CITY EFFECTS      )
```

```
      WRITE(MS,80)
   80 FORMAT( 1H ,20X, 70HCOMBINED EFFECTS. NUMBERS HAVING INDICATED T
     1YPE OF BOTH EFFECTS )
      WRITE( MS,83)
   83 FORMAT( 1H0, 18HRADIATION EFFECT ,17X, 5HFATAL ,32X,7HINJURED,
     1 33X, 9HUNINJURED )
      WRITE( MS,84)
   84 FORMAT( 1H0, 18H    BLAST EFFECT,9X,   5HFATAL, 7X, 7HINJURED,
     119X,9HUNINJURED   ,7X, 5HFATAL, 7X, 7HINJURED, 9X, 9HUNINJURED,
     27X,  5HFATAL ,7X,  7HINJURED, 9X,9HUNINJURED     )
      IF (LSTAGE .NE.1) GO TO 87
      WRITE (MT, 85 )
      WRITE (MT, 86 )
      WRITE (MT, 83 )
      WRITE (MT, 84 )
   87 CONTINUE
      DO 81 K = 1,100
      WRITE(MS,82) K,( SAR(I,K) ,I = 7,15)
   82 FORMAT( 1H0,I3,9HGROUP NO. I ,I3,1H) ,2X,F12.0,2X,F12.0,
     12X, F12.0)
      DO 89 I = 7,15
      EA(I) = 100.*SAR(I,K)/TOTPOP
      EA1(I) = 100. - EA(I)
   89 CONTINUE
      WRITE(MS,88) (EA(I),EA1(I),I = 7,15)
   88 FORMAT(1H ,10X,     3X,F5.1,2X,F5.1,2X,F5.1,1X,1H,
     1                    3X,F5.1,2X,F5.1,2X,F5.1,1X,1H,
     2                    3X,F5.1,2X,F5.1,2X,F5.1 )
      IF (LSTAGE .NE.1) GO TO 69
      WRITE(MT,82) K,( SAR(I,K) ,I = 7,15)
      WRITE(MT,88) (EA(I),EA1(I),I = 7,15)
   69 CONTINUE
   81 CONTINUE
      RETURN
      END
```

MODY WOULD HAVE RESULTED IN BETTER OPTIMIZATION

SUBROUTINE TRDCL (DECK #31)

A. GENERAL

This subroutine computes effects from different weapons on a set of selected tracts for a particular city. It was originally used in program ANDANTE to focus on comparison of blast effects and initial nuclear radiation effects. For each weapon the pressure, thermal radiation, and initial nuclear radiation are computed and listed along with maximum values of these variables and combined weapon injury and survival probabilities.

B. REQUIREMENTS ON CALLING PROGRAM

The requirements on the calling program are basically the same as with the subroutine RADKL. The following common block variables must be defined.

/TMPAND/	NSP - the absolute value of NSP is the number of tracts to be studied. If NSP is positive these tracts are selected in a regular fashion; if negative they are selected at random.
/WPNPRB/	WP - number of weapons
	XZ - weapon E-W location (nmi)
	YX - weapon N-S location (nmi)
/ST44TA/	X - tract E-W location (nmi)
	Y - tract N-S location (nmi)
	POP - tract population
/CITYPR)	NAMEC - city name
	TOTPOP - city total population
	NTRCTS - number of tracts in city
/VULPR/	PSINJ - mean injury overpressure
	D50 - mean lethal dose
	SIG50 - std. dev. of mean lethal dose
	C50 - std. dev. of mean injury dose

SIGC50 - std dev. of mean injury dose

PF - protection factor for initial nuclear radiation

/WPNPR/ DEL - weapon delivery probability

NTYPA - high if burst indicator; see subroutine PROMPT
FLDNA - cute root of weapon yield

/PKPR/ probability of kill/injury vs. distance squared arrays; filled by subroutines FLPKA or FLPKHU.

The subroutine produces listings on the standard output media. The subroutines PROMPT, CUMNOR, and CALRN must be availab.e

C. ALGORITHMS IMPLEMENTED

The distance from each tract studied to each weapon is calculated and effects computed by subroutine PROMPT.

For blast the overall probability of survival, or being uninjured, is the product of the individual probabilities from each weapon. For radiation, the doses are summed and divided by the protection factor. The probability is found from the cumulative normal function. The results are listed for weapons in the order input in /WPNPRB/.

```
      SUBROUTINE TROCL
C     NEVUNS COMPATIBLE
C     LAST REVISED NOV. 17, 1972
C     TO CALCULATE PSI THEM RAD ON A TRACT FROM PRELOCATED WEAPONS
C     ASSUME WEAPON LOCATIONS ARE IN NAUT. MILES ABOUT CITY CG
      COMMON/TOPANG/JRAD,NMENU,IPNCHA,IPUNCH,JPCTO,ADJSTF,LSTAPE,LSTC
     1 ,NSA,DESMR,FRASOP
      COMMON/WPNDAT/IWP,XZ(150),YZ(150),ALT(300)
      COMMON/STADTA/X(6000),Y(6000),POP(6000),V(6000)
      COMMON/CITYPR/NAMEC(20),RLD(2), TOTPOP,RLE(30),NTRCTS
      COMMON/VULPR/BLAC(2),PSIN,,RLA-(5),DSS,SIDSD,CSD,SIGCSD,
     1 RLAB(60),NIPPL, PF, RLBF(10),RLAG(30)
      COMMON/WPNPR/ RLP(2),DEL,RLQ(7), NTYPA,RLQ(13),YLDNA,RLS
      COMMON/POPR/RL1, PKCN(30),DELPSN(30),DSSN(30),DELDSN(30),
     1 PKCI(30),DELPSI(30),DSCI(30),DELCSI(30) ,RLV
      COMMON/EFFCAL/RLI,YLDNU,RLJ,LONG,NTYPE,DIST,RLK,PSI,RLL,THRM,
     1THRMVS, RAD, RLM(13)
      COMMON /TRAN/RNO,RNS
      COMMON/IOPR/RL-,NG,RLAA(10)

      RNO = 4.0
      CALL CALRN
      RNO = 0.
      YLDNU = YLDNA
      NTYPE = NTYPA
C     ASSUME INFINITE VISIBILITY
      THRMVS = 1.
      LONG = 1
      NSP = IABS(NSA)
      RNSP = NSP
      RNTRT = NTRCTS
      DEL = RNTRT/RNSP
      NDEL = DEL
      ITRN = -NDEL + 0
      DO  100 ICL = 1, NSP
      SUMTOT = 1.
      UNINJ = 1.
      PSIMN = 0.
      RADCUM = 0.
      RADMN = 0.
        THRMN = 0.
        THRCM = 0.
      PSYN = 0.
      IF (NSA .LT. 0) GO TO 32
      ITRN = ITRN + NDEL
      GO TO 33
   32 CONTINUE
      CALL CALRN
      TRN = RNTRT*RNO
```

```
      ITRN = IRN
   33 CONTINUE
      XPT = X(ITRN)
      YPT = Y(ITRN)
      POPSP = POP(ITRN)
      WRITE(NO,23)(NAMEC(I),I = 1,20),TOTPOP,NTRCTS
   23 FORMAT( 1H1 ,//,43H INDIVIDUAL TRACT CALCULATIONS FOR CITY OF
     1,20A4,//,15H WITH POPN. OF     ,F10.0 , 8H HAVING    , I5,
     2   7H TRACTS    ,///)
      WRITE(NO,10)ICL,XPT, YPT,POPSP, ITRN
   10 FORMAT(1H0, 9H NO.   ,I6,3X, 7HEAST =    ,F8.4,3X, 8HNORTH = F8.4,
     1 3X, 6HPOP =   ,F8.0,3X,9HTRACT NO.    ,I5)
      WRITE(NO,25)
   25 FORMAT( 1H0, 3HGPN,6H DIST , 9H    PSI   , 9H MAX PSI , 6H  PR  ,
     16H TOTPR, 6H PINJ  , 6H TOTPI  , 7H  RAD   ,7H MAXRAD  ,7H CUMRAD
     26H PRADM  ,6H PRADI   ,
     3 7H  TMRM,  7H ESTMRM, 7H CUMTMR,                   6H SYNRG )
      DO 110 IGPN = 1,100
      TX = XPT - XZ(IGPN)
      TY = YPT - YZ(IGPN)
      SDIST = TX*TX + TY*TY
      DIST = SQRT(SDIST)
      CALL PROMPT
      IF(PSI .GT. PSIMX) PSIMX = PSI
      DD = SDIST
      J = 1
  104 CONTINUE
      J = J + 1
      IF(DSQM(J).GT. DD) GO TO 104
      P    = PMC(J) + (DD - DSQM(J))*DELPSM(J)/DELDSM(J)
      PMO = P
      P = P*PDEL
      J = 1
  106 CONTINUE
      J = J + 1
      IF(DSQI(J).GT. DD) GO TO 106
      P    = PIC(J) + (DD - DSQI(J))*DELPSI(J)/DELDSI(J)
      PIO = P
      SUMTOT = SUMTOT*(1. - PMO)
      PRTOT = 1. - SUMTOT
      UNINJ = UNINJ*(1. - PIO)
      PITOT = 1. - UNINJ
      IF( RAD .GT. RADMX) RADMX = RAD
      RADCUM = RADCUM + RAD
      TVAL = RADCUM/PF
      TVL = (TVAL - D50)/SIGD50
      PRAD = CUMNOR(TVL)
      TINJ = (TVAL - C50)/SIGC50
      PRADI = CUMNOR(TINJ)
      IF(TMRM .GT.TMRMX) TMRMX = TMRM
      TMCM = TMCM + TMRM
      RADP = RAD
      IF(RAD .GT. 99999.) RADP = 99999.
      RADMXP = RADMX
      IF( RADMXP .GT. 99999.)  RADMXP = 99999.
```

```
                RADCUP = RADCUM
                IF( RADCUP .GT. 99999.) RADCUP = 99999.
                THMMP = THMM
                IF(THMM .GT. 99999.) THMMP = 99999.
                THMSP = THMS
                IF( THMS .GT. 99999.) THMSP = 99999.
                THCMP = THCM
                IF( THCM .GT. 99999.) THCMP = 99999.
                IF( PSIMS .GT. 1.25*PSINJ .AND.RADCUM.GT.(C90 + 3)*C90)) PSYN =1.
                WRITE(IWG,20)I*PN,DIST,PSI,PSIMS,PRO,PRTUT, PI0,PITOT, RADP,
               1RADMSP,RADCUP, PHAD,PRACI,THMMP,THMSP,THCMP,PSYN
         20     FORMAT( 1H , I3, F6.2, 3F9.2, 4F6.3, 3F7.1,2F6.3,3F7.1,F6.3)
        110     CONTINUE
        100     CONTINUE
                RETURN
                END
```

SUBROUTINE RDCNTY (DRA #34)

A. GENERAL

This subroutine was developed to input population data and economic data by county to the program ALLEGRO, based upon the National Nodal Network definitions for 1960 census data extrapolated to 1975 as described in IDA Paper P-760. It is retained in the NEVUNS system at least until the 1970 census update of population data is available. It also provides a means of correlation with the earlier data bases.

This subroutine also associates with each county in Area ABM site coverage, and with each city and terminal ABM site coverage.

B. REQUIREMENTS ON CALLING PROGRAM

The calling program must have available input data tapes: The population tape derived from the National Nodal Network, IDA Tape No. , on input tape unit 1; and the county economic tape, IDA Tape No. , on input tape unit 2 if the economic option is used, i.e., switch JECO=1.

If the terminal ABM option is exercised, switch JTABM=1, then city ID number for each defended city must be available in the common block ABMPR. If in addition the terminal defense bypass option is off, JTAVD is not equal 1, then the defense "price" must also be available.

If the area ABM option JAABM is 1 and the area defense avoidance option JAAVD is on, then for each area defense site the site latitude, longitude, cosine of the latitude, and site coverage radius is needed.

PRECEDING PAGE BLANK-NOT FILMED

C. ALGORITHMS IMPLEMENTED

The program reads values of county and city location, population and size and places them in the appropriate common block locations. Economic data is read and stored. If, for some reason the county identifiers are different, an error stop occurs.

The terminal ABM option identifies defended cities and inserts the city price. The area ABM option places a city with the nearest ABM site if the city is under the circular coverage of a site. Other [illegible] options may be implemented if desired and controlled by the parameter JROOT.

The defense options can be used as a mechanism to avoid targeting of certain cities for other purposes, for example, if a set of cities have already been attacked by other weapons.

Documentation of Current IDA Computer Material Developed for DCPA, by Leo A. Schmidt. Unclassified. Institute for Defense Analyses, January 1977, Vol. II--276 pages. (Contract DCPA01-76-C-0213, Work Unit 4126G)

Abstract

This paper is a documentation of computer materials developed by the Institute for Defense Analyses (IDA) for use by the Defense Civil Preparedness Agency (DCPA). All IDA physical data processing materials (IBM cards, magnetic tape, computer printouts) have been surveyed and catalogued. All computer programs are written in FORTRAN (a general knowledge of this language is assumed in the detailed descriptions contained herein). Computer programs considered useful by IDA have been included and documented. A group of general purpose subprograms are described, along with their interfaces with the using programs. Data file formats also have been developed, along with programs for managing these files. Such programs and resulting files are described in detail.

Documentation of Current IDA Computer Material Developed for DCPA, by Leo A. Schmidt. Unclassified. Institute for Defense Analyses, January 1977, Vol. II--276 pages. (Contract DCPA01-76-C-0213, Work Unit 4126G)

Abstract

This paper is a documentation of computer materials developed by the Institute for Defense Analyses (IDA) for use by the Defense Civil Preparedness Agency (DCPA). All IDA physical data processing materials (IBM cards, magnetic tape, computer printouts) have been surveyed and catalogued. All computer programs are written in FORTRAN (a general knowledge of this language is assumed in the detailed descriptions contained herein). Computer programs considered useful by IDA have been included and documented. A group of general purpose subprograms are described, along with their interfaces with the using programs. Data file formats also have been developed, along with programs for managing these files. Such programs and resulting files are described in detail.

Documentation of Current IDA Computer Material Developed for DCPA, by Leo A. Schmidt. Unclassified. Institute for Defense Analyses, January 1977, Vol. II--276 pages. (Contract DCPA01-76-C-0213, Work Unit 4126G)

Abstract

This paper is a documentation of computer materials developed by the Institute for Defense Analyses (IDA) for use by the Defense Civil Preparedness Agency (DCPA). All IDA physical data processing materials (IBM cards, magnetic tape, computer printouts) have been surveyed and catalogued. All computer programs are written in FORTRAN (a general knowledge of this language is assumed in the detailed descriptions contained herein). Computer programs considered useful by IDA have been included and documented. A group of general purpose subprograms are described, along with their interfaces with the using programs. Data file formats also have been developed, along with programs for managing these files. Such programs and resulting files are described in detail.

Documentation of Current IDA Computer Material Developed for DCPA, by Leo A. Schmidt. Unclassified. Institute for Defense Analyses, January 1977, Vol. II--276 pages. (Contract DCPA01-76-C-0213, Work Unit 4126G)

Abstract

This paper is a documentation of computer materials developed by the Institute for Defense Analyses (IDA) for use by the Defense Civil Preparedness Agency (DCPA). All IDA physical data processing materials (IBM cards, magnetic tape, computer printouts) have been surveyed and catalogued. All computer programs are written in FORTRAN (a general knowledge of this language is assumed in the detailed descriptions contained herein). Computer programs considered useful by IDA have been included and documented. A group of general purpose subprograms are described, along with their interfaces with the using programs. Data file formats also have been developed, along with programs for managing these files. Such programs and resulting files are described in detail.

ATE

LME